Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Ingeniería Electromecánica

Diseño y construcción de una pantalla informativa utilizando matrices LED RGB para proyección de texto.

Trabajo de Titulación previo, a obtención del Título de Ingeniero Electromecánico

Autor:

Joffre Fabian Iñiguez Quizhpe

Director:

Dr.C. Jorge Enrique Carrión González, Ph.D.

Loja-Ecuador

2024

Certificación

Loja, 23 de julio del 2024

Dr.C. Jorge Enrique Carrión González, Ph.D.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de titulación denominado: **Diseño y construcción de una pantalla informativa utilizando matrices LED RGB para proyección de texto.**, previo a la obtención del título de **Ingeniero Electromecánico**, de la autoría del estudiante, **Joffre Fabian Iñiguez Quizhpe** con cédula de identidad Nro. **1105177248**, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
JORGE ENRIQUE CARRION GONZALEZ

Dr.C. Jorge Enrique Carrión González, Ph.D.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Joffre Fabian Iñiguez Quizhpe**, declaro ser el autor del presente Trabajo de Titulación y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos de posibles reclamos o acciones legales por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja, la publicación de mi Trabajo de Titulación en el Repositorio Institucional – Biblioteca Virtual.

Firma: 

Autor: Joffre Fabian Iñiguez Quizhpe

Cédula: 1105177248

Dirección: Celi Román - Loja

Fecha: 23 de julio del 2024

Correo Electrónico: joffre.iniguez@unl.edu.ec

Teléfono: 0999576854

Carta de autorización por parte del autor para la consulta, reproducción parcial o total, y/o publicación electrónica de texto completo, del Trabajo de Titulación.

Yo, **Joffre Fabian Iñiguez Quizhpe**, declaro ser autor del Trabajo de Titulación denominado: **Diseño y construcción de una pantalla informativa utilizando matrices LED RGB para proyección de texto.**, como requisito para optar por el título de: **Ingeniero Electromecánico**, autorizo al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido de la siguiente manera en el Repositorio Institucional. Los usuarios pueden consultar de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Titulación que realice un tercero.

Para la constancia de esta autorización en la ciudad de Loja, los veintitrés días del mes de julio del dos mil veinticuatro.

Firma: 

Autor: Joffre Fabian Iñiguez Quizhpe
Cédula: 1105177248
Dirección: Celi Román - Loja
Fecha: 23 de julio del 2024
Correo Electrónico: joffre.iniguez@unl.edu.ec
Teléfono: 0999576854

DATOS COMPLEMENTARIOS:

Director del trabajo de titulación: Dr.C. Jorge Enrique Carrión González, Ph.D.

Dedicatoria

El presente Trabajo de Titulación, quiero dedicarlo principalmente a Dios quien ha sido mi guía y fortaleza, con nostalgia y mucho amor a mis padres Joffre Iñiguez y Mary Quizhpe quienes pusieron su confianza y apoyo incondicional en mi formación académica y personal.

A mi amigo Ermel por estar en todo el camino recorrido en el ámbito universitario. A mis mentores que han sabido guiarme y formarme con los pensamientos correctos.

Joffre Fabian Iñiguez

Agradecimiento

Retribuirme por el tiempo dedicado a cumplir este objetivo. Luego agradezco a mis padres por el sacrificio excepcional que han realizado por mis estudios y la culminación de mi carrera universitaria. Un reconocimiento importante a la Universidad Nacional de Loja; a la Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables, y a la carrera de Ingeniería Electromecánica, por permitirme formar profesionalmente en esta gran institución.

Joffre Fabian Iñiguez

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Diseño y construcción de una pantalla informativa utilizando matrices LED RGB para proyección de texto.

2. Resumen

El presente trabajo de titulación está enfocado en el desarrollo de un prototipo de pantalla informativa con el uso de tecnología LED RGB. El objetivo principal fue la ejemplificación y desarrollo de un sistema capaz de proyectar texto alfanumérico e imágenes, con aplicaciones potenciales en diversos entornos informativos que se consolida como una propuesta tecnológica para la difusión de información. Se indagó sobre tecnologías de control para matrices LED RGB y se presenta un método de matriz de ponderación para seleccionar la tecnología LED y el controlador óptimo para el prototipo. Se determinó que las matrices WS2812B son las adecuadas debido a su capacidad de control individual de LEDs, su alta luminosidad y la flexibilidad en la creación de display's escalables. En lo que respecta al controlador, la selección de Arduino Due y ESP32 como hardware de control es debido a sus características avanzadas y capacidad de procesamiento, lo que permite gestionar de manera eficiente la comunicación y el control de las matrices LED; además, permitió contrastar su funcionalidad configurando la pantalla en dos formatos, horizontal y vertical. Las principales conclusiones señalan que las pruebas realizadas al prototipo de pantalla confirman su capacidad para proyectar texto e imágenes con la resolución propuesta de 64 x 48 pixeles, pero es posible expandir el sistema agregando más matrices de visualización para mejorar la calidad y claridad de los mensajes mostrados. A pesar de esta limitación, el prototipo demostró ser una plataforma efectiva para comprender el funcionamiento de la tecnología LED RGB.

Palabras claves: Arduino DUE, ESP32, LED, pantalla, ws2812b.

Abstract

The present thesis work focuses on the development of an informational display prototype using RGB LED technology. The main objective was to exemplify and develop a system capable of projecting alphanumeric text and images, with potential applications in various informative environments, establishing itself as a technological proposal for information dissemination. The investigation explored control technologies for RGB LED matrices, presenting a matrix weighting method to select the optimal LED technology and controller for the prototype. It was determined that WS2812B matrices are suitable due to their capability for individual LED control, high luminosity, and flexibility in creating scalable displays. Regarding the controller, the selection of Arduino Due and ESP32 as control hardware was based on their advanced features and processing capacity, enabling efficient management of communication and LED matrix control. Furthermore, it allowed contrasting functionality by configuring the display in both horizontal and vertical formats. The main conclusions indicate that tests conducted on the prototype confirm its ability to project text and images at the proposed resolution of 64 x 48 pixels. However, expanding the system by adding more display matrices could enhance the quality and clarity of displayed messages. Despite this limitation, the prototype proved to be an effective platform for understanding the operation of RGB LED technology.

Keywords: Arduino Due, ESP32, LED, display, WS2812B.

3. Introducción

Las pantallas LED se caracterizan por presentar información y publicidad la cual puede ser sustituida en intervalos de tiempo cortos. La tecnología LED RGB se ofrece como una solución óptima al presentar su uso en pantallas, paneles o letreros electrónicos en diferentes formas, tamaños, capacidades y diseños, inclusive los juegos de luces que utilizan para atraer a un posible cliente y por otro lado en el ámbito educativo para fortalecer la comunicación y entrega de información en el ambiente institucional. Estos dispositivos electrónicos de LED´s hacen una parte muy visible y crítica de cómo un negocio puede permanecer en contacto con sus clientes sobre una base cotidiana, con sus últimos mensajes de publicidad y ventas.

A pesar de que en los últimos años se ha masificado el uso de la tecnología LED RGB, está tecnología necesita de firmware especializado para la programación de la información a comunicar en el panel LED. Siendo necesario identificar métodos y modelos que permitan diseñar y construir un prototipo de matrices LED RGB para proyección de texto. La investigación contribuye a la comprensión del funcionamiento de las técnicas de LED RGB para crear carteles informativos escalables de bajo costo, además, permitirá conocer las formas de integrar esta tecnología para un uso específico.

La metodología empleada en este proyecto se dividió en varias etapas para asegurar un análisis y desarrollo íntegro del prototipo de pantalla informativa. En primer lugar, se identificaron los requisitos específicos para el sistema como son el tamaño y resolución de la pantalla; lo que incluyó las especificaciones técnicas (voltaje de operación, controladores, tipos de matriz LED) y funcionales (tipo de contenido, brillo ajustable, frecuencia de actualización, entrada de usuario). A continuación, se buscó información sobre las tecnologías LED RGB disponibles, seleccionando las matrices WS2812B por sus características favorables como son los protocolos de transferencia de datos seriales, versatilidad y escalabilidad. Posteriormente, se procedió al dimensionamiento de los componentes y materiales, seleccionando Arduino Due y ESP32 como controladoras por su capacidad de procesamiento y flexibilidad. Se diseñó y construyó un prototipo modular, que fue sometido a varias pruebas para evaluar su desempeño en la proyección de texto e imágenes. Finalmente, se documentaron los resultados del análisis y las pruebas, identificando áreas de mejora como la resolución de las matrices LED.

Para cumplir con el tema establecido se plantearon los siguientes objetivos a cumplir en el desarrollo de este proyecto.

3.1 OBJETIVOS DE INVESTIGACIÓN

Para este proyecto se plantean los siguientes objetivos:

Objetivo general.

Construir un prototipo de pantalla informativa utilizando matrices LED´s RGB para proyectar texto alfanumérico e imágenes.

Objetivos específicos.

- Analizar tecnologías LED RGB para la construcción de pantallas informativas con matrices LED.
- Dimensionar componentes y materiales para la pantalla informativa escalable con tecnologías LED RGB.
- Validar el prototipo de pantalla informativa utilizando matrices LED RGB para proyectar texto alfanumérico e imágenes.

En cuanto al contenido del trabajo de titulación, la primera parte consta de un breve estado del arte y un componente teórico, fruto de la búsqueda y recopilación de información, donde se enfoca en puntos importantes como: la teoría de la imagen, al igual que el análisis y aplicación de la tecnología LED para la proyección visual. Tipos de tecnologías LED RGB aplicadas a pantallas informativas; de igual manera se presenta la descripción de los diferentes partes que componen este tipo de pantallas, así como la fundamentación para el dimensionamiento y selección de las diferentes partes que contribuyen a la ejecución de la investigación. La segunda parte del proyecto, trata sobre los materiales y la metodología que se ha utilizado para poder lograr los objetivos planteados.

Seguidamente en la tercera parte de la investigación se exponen los diferentes resultados obtenidos, iniciando con la obtención del tipo de LED RGB a través de la matriz de ponderación, seguido de la obtención del diseño y dimensiones como referencia para el prototipo, para posteriormente ir dimensionando las partes de los diferentes sistemas como; el sistema de alimentación, sistema de control de las matrices LED y la estructura encargada de soportar todos los elementos. Luego de presentar la programación necesaria para controlar las matrices LED y poder proyectar texto alfanumérico o a su vez imágenes.

Finalmente, luego de la obtención del prototipo se expone la discusión, seguido de las conclusiones, y las respectivas recomendaciones sobre los resultados obtenidos en este trabajo.

4. Marco teórico

4.1 Capítulo I

Teoría de la imagen

En esta sección se realiza un análisis breve de los conceptos de imagen, se explican las características más importantes:

4.1.1 Noción general de imagen

Moles (2001) señala que "Una imagen es el soporte de la comunicación visual, que materializa un fragmento del entorno óptico (universo perceptivo), susceptible de subsistir a través del tiempo y que constituye uno de los componentes principales de los medios masivos de comunicación (fotografía, pintura, ilustraciones, esculturas, cine, televisión)".

De forma general, se entiende por *imagen* la apariencia visible de una figura, así una imagen es la representación de un objeto determinado.

4.1.2 La imagen como señal

Suárez (2013) indica que "Una señal es una función asociada a un fenómeno físico cuya variación determinada en un dominio dado porta información codificada. Las imágenes como tal, una vez representadas son 2D (2 dimensiones) y las señales 1D (una dimensión)" (p. 8).

Además, señala que el ámbito en que se mueve una señal se denomina dominio, en el caso de sistemas ópticos el dominio es espacial, la función es bidimensional y el término *señal* suele confundirse con *imagen* que, es una función asociada a una distribución de intensidades de luz en un determinado dominio espacial. Una imagen se reduce a una señal lineal que recorre un canal de determinadas características.

4.1.3 Resolución y densidad de píxeles

Las imágenes se pueden representar mediante un conjunto apilado de celdas a las que se asignan valores. Cada una de las celdas de dicho conjunto se llama píxel. Por lo tanto, el pixel es un elemento más pequeño y en un gran conjunto representan una imagen, como se observa en la **Figura 1**. Un píxel es un concepto ligero que no posee una medida definida, es una unidad de información. No se puede afirmar si un pixel mide 1 mm o 1 m ya que no se corresponde con un tamaño concreto, en principio, es solamente una medida de división en celdas. De este modo, se puede hablar de una imagen que tenga 150 x 120 píxeles sin saber qué tamaño real y físico tiene, como referencia se puede mencionar que se ha dividido la imagen en 18000 píxeles (celdas).

Figura 1. Representación de La Gioconda de Leonardo da Vinci en una imagen de 64x82 píxeles.
Fuente: *(PipBricks, 2023).*

Sin embargo, cuando se le asigna una *resolución* a esa imagen, por ejemplo, si se menciona que una imagen tiene 100 píxeles por pulgada, esto hace referencia a que cada 2,54 cm, habrá 100 celdas, con lo que cada píxel equivaldrá a 2,54 mm. La resolución de la pantalla se refiere al número total de píxeles que puede mostrar. En el caso de las matrices LEDs RGB, esto está determinado por el número de LEDs en la pantalla y su disposición. Una mayor densidad de píxeles (más LEDs por unidad de área) resulta en una imagen más nítida y detallada.

4.1.4 Brillo y contraste

El brillo se refiere a la intensidad luminosa de la pantalla, mientras que el contraste se relaciona con la diferencia entre las áreas más claras y más oscuras de la imagen. Es importante ajustar el brillo y el contraste de la pantalla para garantizar una visualización óptima en diferentes condiciones de iluminación.

4.1.5 Color y saturación

La luz tiene diferentes colores en función de la longitud de onda. A las ondas que son visibles, se las denomina espectro visible. Dentro de ese espectro visible no se distingue el ultravioleta y el infrarrojo pues se encuentran por debajo y por encima de la capacidad visual. Además, el espectro de la luz está comprendido entre los 380 y los 720 nanómetros. De este espectro resultan una gama de colores fruto de la descomposición del blanco (Solas, 2014).

La tecnología de matrices LED RGB permite la reproducción de una amplia gama de colores. Es importante seleccionar colores adecuados y controlar la saturación para evitar distorsiones cromáticas y garantizar una representación precisa del contenido.

4.1.6 Perspectiva y visualización

La distancia y el ángulo de visión afectan la percepción de la imagen en la pantalla. Es importante considerar la ubicación de la audiencia y el ángulo de visión al diseñar el contenido para garantizar una visualización clara y legible desde diferentes posiciones.

4.1.7 Composición y diseño visual

La composición se refiere a la disposición y organización de los elementos visuales en la pantalla. Es importante utilizar principios de diseño visual, como la alineación, el equilibrio y la jerarquía, para crear contenido fácil de entender.

4.1.8 Movimiento y animación

El uso de efectos de movimiento y animación puede captar la atención del espectador y transmitir información de manera dinámica. Es importante utilizar el movimiento con moderación y asegurarse de que sea relevante para el contenido presentado.

4.1.9 Legibilidad y tipografía

La legibilidad se refiere a la facilidad con la que se pueden leer los textos en la pantalla. Es importante seleccionar fuentes de texto adecuadas y asegurarse de que el tamaño y el contraste sean suficientes para garantizar una legibilidad óptima, incluso a distancia.

4.1.10 Interactividad y participación

La interactividad permite a los espectadores participar activamente con la pantalla, lo que puede aumentar el compromiso y la efectividad de la comunicación. Es importante incorporar elementos interactivos de manera intuitiva y proporcionar retroalimentación clara al usuario.

4.1.11 Cuadros y persistencia de la visión

Se puede describir un cuadro como la representación visual que se muestra en una pantalla para el espectador, pudiendo ser tanto caracteres como dibujos. En el tipo de pantallas con LEDs RGB los videos se muestran al exponer rápidamente una serie de cuadros para que el espectador no note ninguna interrupción. La rapidez con la que se actualizan estos cuadros se llama frecuencia de refresco. Si esta frecuencia es superior a un cierto límite, el observador no percibirá ninguna interrupción. Para las pantallas de LED, se sugiere una frecuencia de refresco de al menos 60 Hz.

Soto, Soto, & Vásquez (2012) señalan que “La persistencia de la visión es el fenómeno del ojo humano que permite a las imágenes de video ser vistas sin parpadeo, cuando el sistema de visión humano está presenciando una imagen, esta imagen continúa siendo percibida uniformemente por el pensamiento y se queda retenida en el campo visual del observador por un corto tiempo. Este fenómeno permite un video libre de parpadeos y discontinuidades. (p. 4.)

4.2 Capítulo II

Generalidades de la tecnología LED

4.2.1 Diodo emisor de luz

Gago (2012) define el diodo emisor de luz o LED (Light Emitting Diode) como un dispositivo semiconductor que emite luz con una longitud de onda monocromática especifica muy bien definida cuando se polariza de forma directa pasando una corriente eléctrica entre sus extremos.

Actualmente los diodos luminosos permiten crear diseños creativos para conseguir soluciones luminosas innovadoras con la variedad de colores de los LEDs, su reducido tamaño y la flexibilidad de los módulos. Las partes de un diodo LED se observan en la **Figura 2**.

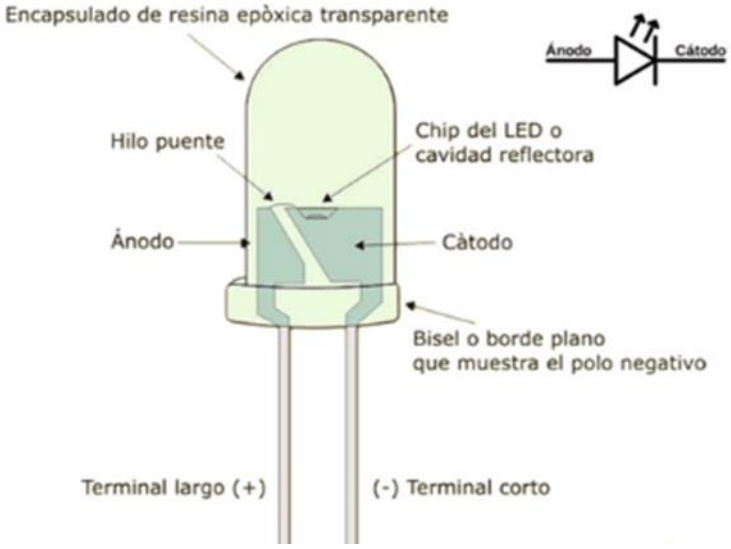


Figura 2. Partes diodo emisor de luz (LED).
Fuente: (TALLERELECTRONICA.COM/BLOG, 2023).

4.2.1.1 Ventajas en la aplicativa del LED

De acuerdo con MegaLámparas (2023), algunas de las ventajas que tiene el uso del diodo LED, son:

i. *Ahorro Energético:* Los diodos LED proporcionan un ahorro energético entre un 80 y 90% comparado a la iluminación tradicional. Esto sucede porque el 80% de la energía que consume un LED se transforma en luz, a diferencia de las bombillas tradicionales que transforman la mayor parte de la energía en calor, necesitando, en consecuencia, más energía para poder emitir luz.

ii. Vida útil: Una bombilla tradicional puede tener un tiempo promedio de uso de 5000 horas, una bombilla LED será aproximadamente de 100.000 horas. Los

diodos LED no se averían ni dejan de funcionar de repente, sino que van disminuyendo su intensidad de iluminación de forma decreciente.

iii. Encendido inmediato: Como los diodos LED no necesitan calor para su funcionamiento, pueden alcanzar su máximo rendimiento de forma inmediata y al 100%.

iv. Costo de mantenimiento e instalación bajo: Debido a la larga vida que tienen los diodos LED su costo de mantenimiento es bastante bajo, pues no necesitan ser sustituidos en un largo tiempo. De igual forma, la instalación de este tipo de iluminación tiene un costo más bajo que el de la iluminación tradicional.

v. Los LED pueden producir luz de un color específico, sin la necesidad de utilizar filtros adicionales, lo que ahorra peso y los hace más eficientes, así como la actual variación y generación de múltiples colores, basado esto en la tecnología RGB lo cual crea posibilidad de diseños creativos considerando la gran variedad de colores, medidas compactas y flexibilidad de los módulos.

4.2.1.2 Desventajas aplicativas del LED

Aparte de brindar grandes beneficios, también posee una gran desventaja:

i. Su emisión de iluminación es baja mediante su misma fuente angular de visibilidad entre los 30 y 60 grados.

ii. Para conseguir la iluminación de una bombilla tradicional se necesitan tres LED.

4.2.2 Denominación RGB

Los LED RGB consisten en una combinación de un LED rojo, uno azul y otro verde (RED, GREEN, BLUE), cómo se expone en la **Figura 3**. Ajustando independientemente la tonalidad pasada como parámetro por cada uno de ellos, los LED RGB son capaces de producir una amplia gama de colores.

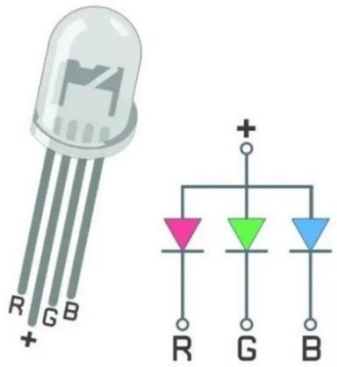


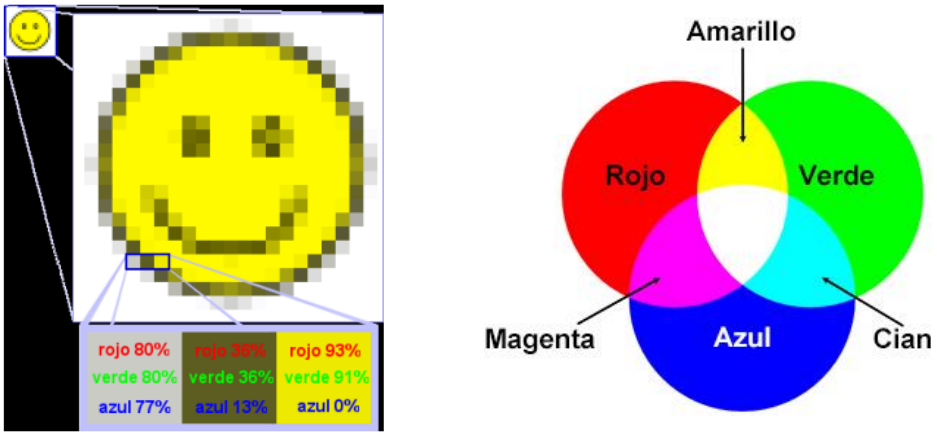
Figura 3. Diodo LED RGB.
Fuente: Anónimo.

Los colores para un LED RGB se expresan mediante números que van desde 0 hasta 255. Para crear el color rojo, el valor máximo se asigna al rojo y el mínimo a los otros colores. Por lo tanto, el rojo se representa como $R = 255; G = 0; B = 0$, y este patrón se aplica a los demás colores de manera similar.

La definición de colores se basa en el principio de la mezcla aditiva de colores **(Figura 4).** Debido a que internamente los tres LED están ubicados muy juntos, el ojo humano no puede distinguirlos, ya que solo ve la mezcla de color final.

4.2.2.1 Aplicaciones del LED RGB

Para una pantalla básica un LED RGB puede ser usado como un píxel. En la **Figura 4** se aprecia un ejemplo de píxeles y canales de color. La imagen original e inicial a representar es la figura de la esquina superior izquierda. En la versión ampliada de la cara para una pantalla, se puede ver los pixeles individuales de la imagen que se disponen en filas y columnas para conforman la imagen. Además, se puede apreciar tres píxeles aún más ampliados, en estos se indica los valores para cada canal rojo, verde y azul escritos sobre cada píxel.



a) *b)*

Figura 4. *a)* Valor de cada color para un píxel en la imagen de un emoji,
b) Espacio de color RGB.
Fuente: *a)* (Suárez, 2013), *b)* (ART ROCKET, 2023).

4.2.3 Matrices de LEDs

Una matriz de LEDs es un conjunto o arreglo de LEDs dispuestos de manera que forman filas y columnas, pueden ser encendidos y apagados individualmente desde un microcontrolador. Se clasifican como una pantalla de pocos píxeles, en los cuales se puede representar gráficos y textos, tanto estáticos como en movimiento.

La **Figura 5** muestra dos configuraciones diferentes de una matriz 4x4, la diferencia está en el método que es utilizado para controlar los LEDs. En la configuración en ánodo

común, la corriente va hacia los puertos 1 a 4 y en la configuración en cátodo común, la corriente va desde los puertos 1 a 4.

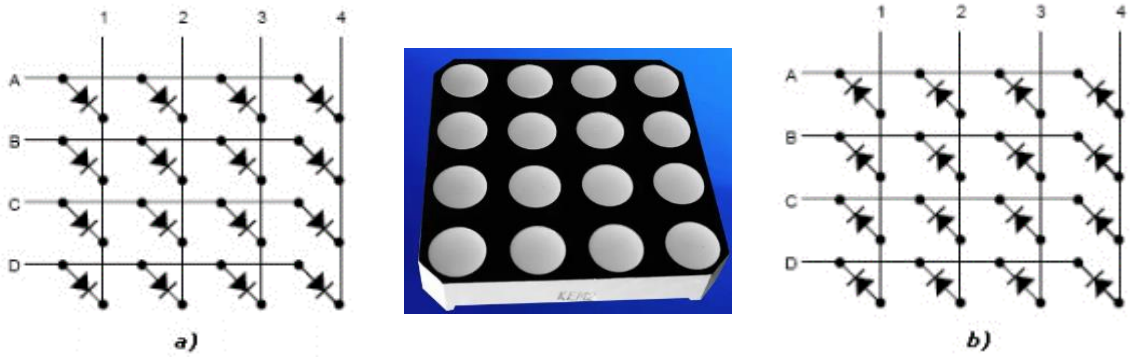


Figura 5. Arreglo matricial de *a)* ánodo común (por fila) *b)* cátodo común (por fila).
Fuente: (Soto, Soto, & Vásquez).

Actualmente existen matrices de punto RGB de mayor tamaño, como se observa en la **Figura 6**.

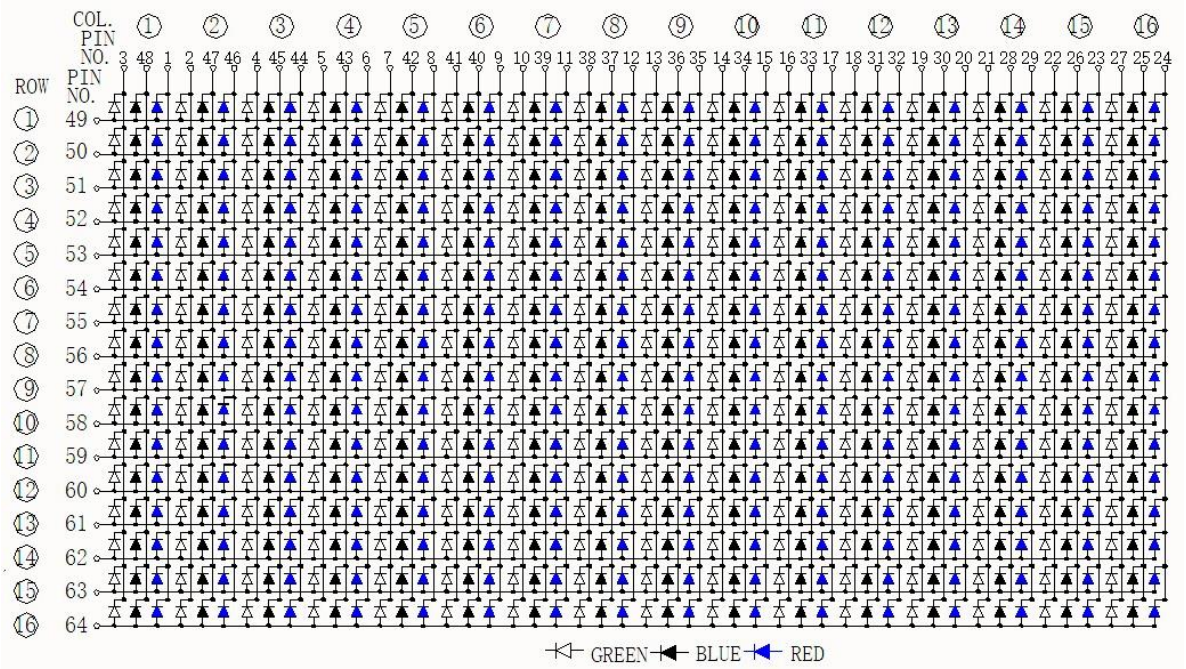


Figura 6. Diagrama de circuito interno de una matriz LED RGB 16x16.
Fuente: (Houkem, 2023)

4.2.4 Control de matrices LEDs

Utilizar varios LEDs individualmente es relativamente simple, sin embargo, cuando el número de LEDs incrementa, los recursos que se necesitan para operarlos resultan bastante extensos. El control de numerosos LEDs a la vez se puede realizar de diferentes formas, una de ellas es el multiplexado de una matriz LED.

4.2.4.1 Multiplexación de una Matriz de LEDs

Este método se utiliza normalmente para pequeñas matrices y cubos de LEDs; aunque se usa en la actualidad, es ineficiente, inflexible y muy complicado de fabricar. En grandes proyectos puede resultar de gran complicación. Este método consiste en encender a velocidades extremadamente rápidas (no detectables por el ojo humano) un LED tras otro (generalmente solo se enciende uno a la vez) en forma de pulso. De esta forma se puede dibujar patrones sobre las matrices, dando la ilusión de que varios LEDs están encendidos a la vez (Harwood, 2017).

Por multiplexaje, solo una fila de la matriz de LEDs es activada en un intervalo de tiempo, este método se aplica porque un terminal del LED (sea el ánodo o el cátodo) está unido a una sola fila. Para comprender este funcionamiento cómo se observa en la **Figura 7**, si energizamos A y B al mismo tiempo, será imposible direccionar un LED individualmente dentro de estas dos filas. Por ejemplo, si la línea 1 conduce cuando A+B conduce, dos LEDs se encenderán simultáneamente

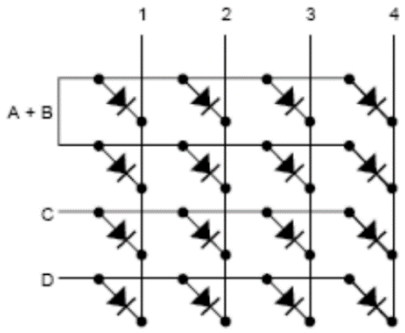
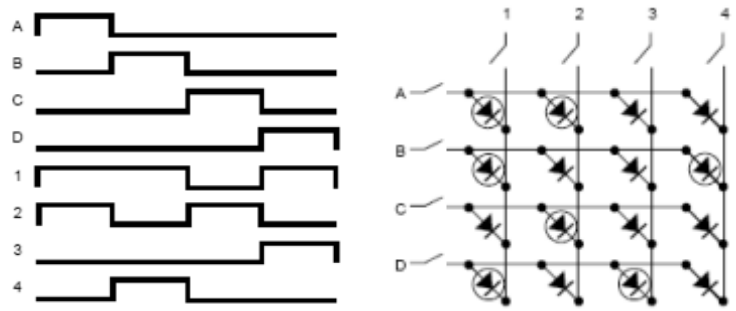


Figura 7. Ejemplo de conexión de matriz LED.
Fuente: (Soto, Soto, & Vásquez).

Para ello se emplea la técnica de multiplexaje, ya que permite establecer un orden de encendido y apagado de los LEDs dentro de la matriz, así también controlar los intervalos de tiempo que deben encender y apagarse cada LED. El manejo paralelo de LEDs es inapropiado debido a las corrientes parásitas. Este fenómeno ocurre si la resistencia dinámica de los LEDs en paralelo difiere por mucho. La multiplexación por división de tiempo se la realiza en orden secuencial (A hasta D).

Solamente una fila es energizada en un único instante de tiempo, durante el período en el cual una fila es energizada, los LEDs deseados son encendidos, energizando las columnas apropiadas y respectivas de acuerdo a los datos que se desean mostrar. Este proceso es conocido también como barrido.



a) *b)*

Figura 8. Definición de multiplexado de una matriz de LEDs. La parte *a)* es un diagrama de tiempos y muestra cuando y cual interruptor son presionados. Los círculos en la parte *b)* indican cuales LEDs son encendidos cuando la secuencia es desarrollada.
Fuente: *(Soto, Soto, & Vásquez)*.

En la **Figura 8** se expone solamente una sección de una matriz. El esquema de control mostrado en esta figura puede ser aplicable también a arreglos muy grandes de LEDs. El tamaño máximo depende de la tasa máxima a la cuál es factible distribuir y procesar datos.

4.2.4.2 LEDs inteligentes.

Otra forma de controlar LEDs, es gracias al LED inteligente, consiste en tiras de estos que tienen un microchip único en cada LED que le permiten controlarlos individualmente o en grupos. La lógica integrada recibe la señal necesaria y la traduce iluminando su correspondiente LED en el color identificado. De esta forma se puede colocar un LED tras otro en forma de cascada y que la señal se vaya enviando de un microchip a otro. La principal diferencia en este método es que el LED no está realmente parpadeando, sino que se mantiene encendido mientras el resto de LEDs se van encendiendo.

Algunos modelos incorporan este microchip dentro del mismo componente LED, de esta manera el tamaño del componente queda considerablemente reducido, ventaja a tener en cuenta a la hora de crear una superficie de alta densidad de pixeles. Estos LEDs son denominados *addressable led* y existen dos tipos que se describen a continuación:

- Con cuatro conexiones en la entrada

  (5V, GND, señal de datos y señal de reloj)
- Con tres conexiones en la entrada.

  (5V, GND y señal de datos)

Los modelos más conocidos en ambos casos son:

Con señal de reloj se encuentran los modelos: WS2801, WS2803, LPD6803, LPD8806, SMD16716, SMD5050, APA102.

Sin reloj se encuentran los modelos: WS2812, WS2812B, P9813, WS2813, TM1803, TM1804, TM1903, UCS1903, SK6812.

4.3 Capítulo III

Tipos de tecnologías para pantallas publicitarias

La publicidad basada en pantallas publicitarias es una de las más utilizadas por empresas, tanto para promociones de interior como de exterior. La flexibilidad para configurar todo tipo de mensajes, el dinamismo de estos y la variedad de tamaños ha provocado toda una revolución que, incluso, ha cambiado el aspecto de muchas calles y avenidas en grandes ciudades.

4.3.1 Pantallas LED

Son elementos basados en diodos o LED encargados de emitir luz. Esos pequeños diodos se agrupan en pixeles para poder crear imágenes, videos y textos.

Los tipos más comunes en aplicaciones de pantalla LED son los paneles TN (*Twisted Nematic*) o VA (*Vertical Alignment*) que se exponen en la **Figura 9**.

4.3.1.1 Paneles LED

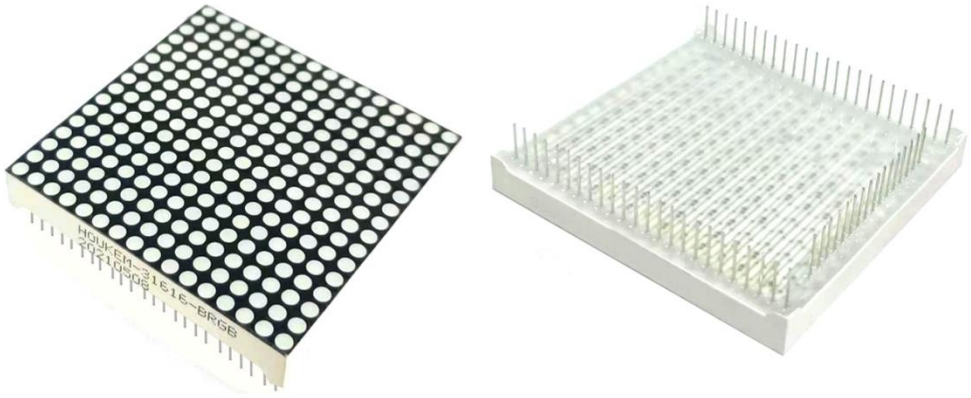
Estos paneles prefabricados para construir pantallas informativas o de publicidad difieren en el tamaño de dígito/alfa/matriz y puede variar de 0,03 a 0,4 pulgadas, siendo los tamaños más comunes 0,2, 0,3 y 0,4 pulgadas. Para denominar estos paneles se indica si es para uso en interiores o exteriores además de la nomenclatura P seguida de un número. En el caso de la **Figura 9**, se trata de un panel para interiores con un paso de 3,9 mm entre cada pixel LED.

Figura 9. Panel LED para interiores SMD P3.91.
Fuente: (DINALIGHT, 2023)

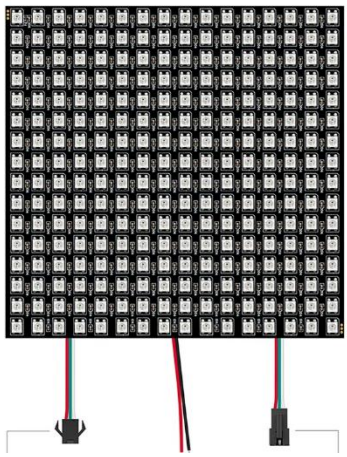
4.3.1.2 Matrices LED

La tecnología de matrices LED RGB ha experimentado un notable avance en las últimas décadas, permitiendo la creación de pantallas informativas personalizables. Dentro de este grupo existen las matrices de LEDs RGB y matrices con los LEDs inteligentes para proyección

de imagen; en el primer caso cada LED tiene una conexión individual mientras que en la matriz de LEDs inteligente corresponde a una conexión en serie de los mismos, cómo se expone en la **Figura 10**. Además, existen otros modelos que corresponden a un tipo de barra de luz, siete segmentos y matriz de puntos, en el se pueden observar muchos de los modelos existentes.



a)



b)

Figura 10. *a)* Matriz LED RGB 31616-A 16x16, *b)* Matriz LED WS2812B 16x16.
Fuente: *a)* (Houkem, 2023), *b)* (Alibaba, 2023)**.**

4.3.2 Pantallas LCD

Poseen diodos para emitir la luz, pero las imágenes no se forman por pixels sino por cristal líquido. Este tipo de tecnología admite todo tipo de colores y, aunque admite oscuros es conocido por ser un recurso publicitario muy luminoso (Mocholí, Borja, & Sánchez, 2023) cómo se expone en la **Figura 11**.

En las pantallas de cristal loquido (LCD) su uso comercial está limitado al tamaño, de hecho se reducen al tamaño máximo de monitores y televisores. Por esta razón, suelen ser recursos publicitarios para interior: pantallas en vagones y estaciones de metro; centros comerciales, restaurantes, etc.

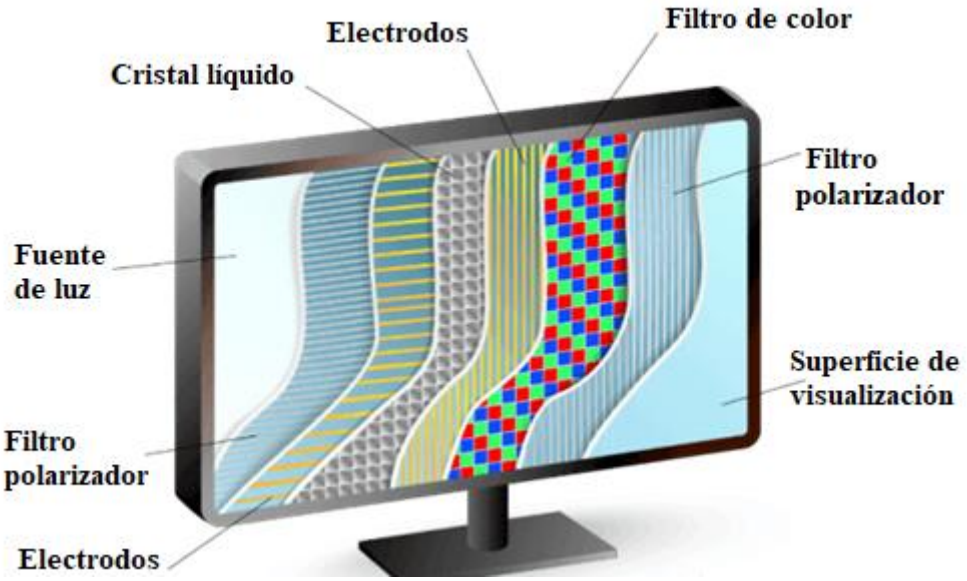


Figura 11. Conjunto de pantalla LCD.
Fuente: (Orient Display, 2023).

4.3.3 Pantallas OLED

Mocholí, Borja, & Sánchez (2023) señalan que "las pantallas OLED cuentan con diodos para emitir la luz peros son orgánicos, de carbono y no necesitan de cristal líquido ya que el propio carbono es capaz de crear una capa de pixels que emite las imágenes deseadas según las estimulaciones eléctricas" (p. 3)

Como ocurre con las pantallas LCD, su producción es costosa, por lo que no se utilizan para pantallas de gran formato (**Figura 12**).

Figura 12. Diferencia pantalla LCD vs OLED.
Fuente: (Lenovo, 2023).

4.4 Capítulo IV

Componentes de la pantalla LED RGB

El diseño y desarrollo del prototipo de pantalla informativa se basa en la selección de materiales y componentes para garantizar un rendimiento adecuado en cada parte que conforma una pantalla de este tipo.

4.4.1 Módulo LED

Entre las diversas opciones disponibles en el mercado, las matrices LED WS2812B que se presentan en la **Figura 13** se destacan por su facilidad de uso, versatilidad y escalabilidad. Estas matrices, que integran LEDs RGB controlables individualmente, ofrecen algunas posibilidades para la creación de efectos visuales dinámicos y la visualización de información en tiempo real.

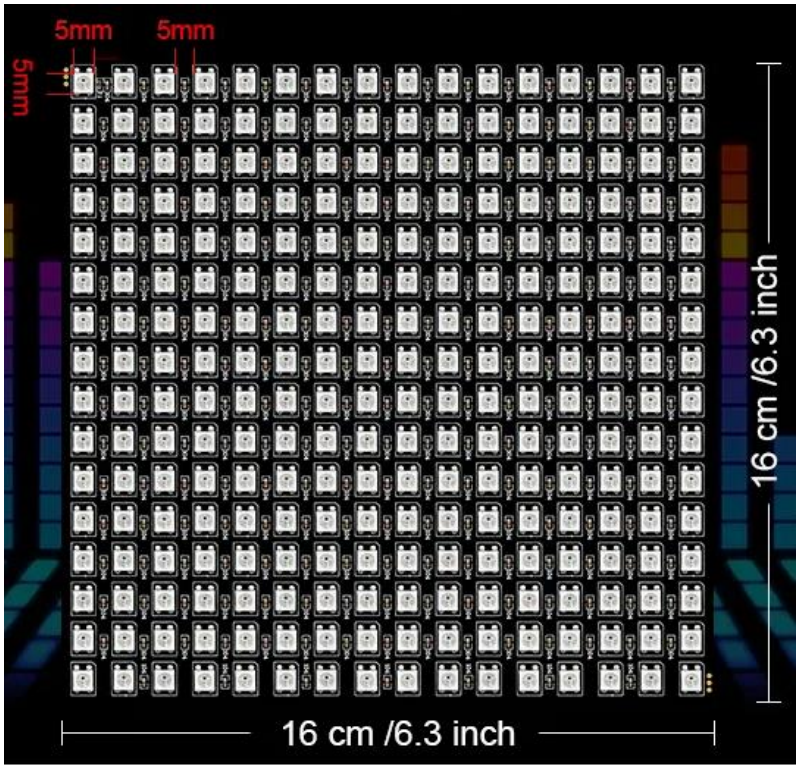


Figura 13. Representación de una matriz LED WS2812B 16x16 cm.
Fuente: (Alibaba, 2023)

Las especificaciones técnicas principales de las matrices WS2812B de esta matriz se presentan en la **Tabla 1**.

Tabla 1. Características de la matriz LED WS2812B.

Especificaciones			
Modelo de chip LED	SMD5050	Vida promedio (horas)	50000
Fuente de alimentación	Corriente continua (DC)	Voltaje	5V
Número de LED/Matriz	256	Potencia	0,3 W/píxel
Impermeable	No	Dimensiones	16 cm * 16 cm

Fuente: (Alibaba, 2023)

A continuación, se explica el funcionamiento de la tecnología LED WS2812B.

4.4.1.1 Tecnología WS2812B

Son LEDs RGB que disponen de lógica integrada, en la **Figura 14** se expone este tipo de tecnología, poseen un chip controlador encapsulado en un pequeño paquete de montaje en superficie controlado a través de un solo cable. Se pueden usar individualmente, encadenados en cadenas más largas.

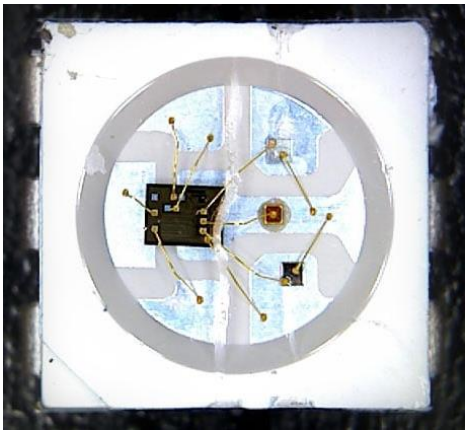


Figura 14. Vista microscópica para apreciar componentes del LED WS2812B.
Fuente: (adafruit, 2023).

El circuito de control incluye una memoria tipo LATCH inteligente (circuito electrónico que almacena información en sistemas lógicos asíncronos), un módulo de acondicionamiento y amplificado de señal, un oscilador de precisión interno y un circuito regulador de voltaje para asegurar el color y el brillo del LED, las características y beneficios de LED WS2812B se presentan en la **Tabla 2**.

Tabla 2. Características del LED WS2812B.

1	Protección inteligente contra conexión inversa.
2	El circuito de control y el LED comparten la fuente de alimentación.
3	El circuito de control y el LED están integrados en un solo componente.
4	Circuito de condicionamiento y amplificación de señal incorporado.
5	Un LED RGB es capaz de mostrar 16777216 colores.
6	Transmisión de datos en serie.
7	Velocidad de transferencia de datos de 800kbps.

Fuente: *(Vazquez, 2019).*

El LED WS2812B utiliza un modo único de NZR (sin retorno cero) como protocolo de comunicación para la transferencia de datos. Después de que el LED enciende, el pin DIN recibe datos del microcontrolador, el primer LED almacena sus respectivos datos y los envía a la memoria LATCH interna, los datos restantes pasan al circuito de acondicionamiento y amplificación de señal y son enviados al siguiente LED de la serie a través del pin DOUT, todo este proceso se puede ver representado en la **Figura 15**.

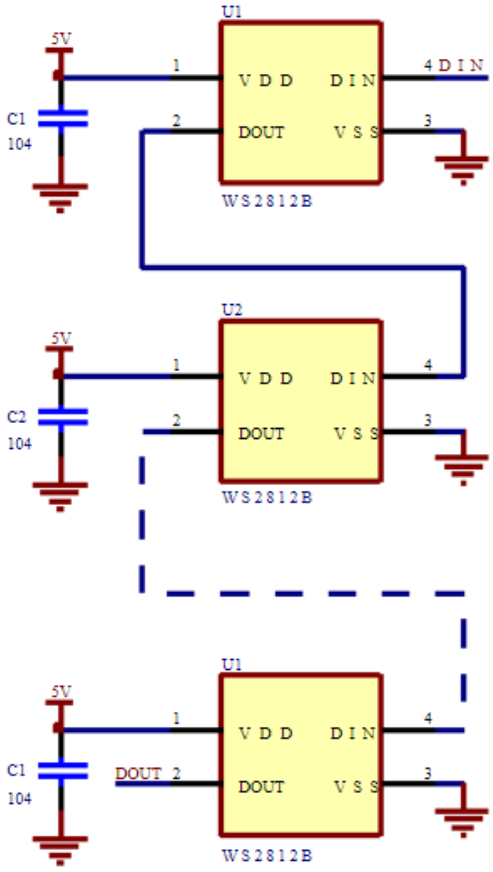


Figura 15. Aplicación de LEDs WS2812B.
Fuente: (Worldsemi, 2023).

Para encender un LED WS2812B son necesarios 24 bits de datos. Estos 24 bits contienen los niveles de brillo que tendrá el LED RGB (8 bits para el rojo, 8 bits para el verde y 8 bits para el azul), logrando que el LED RGB sea capaz de mostrar más de 16 millones de colores, como se observa en la **Figura 16**. La cadena de 24 bits de datos empieza por el bit más significativo para el nivel de brillo correspondiente al color verde y termina con el bit menos significativo del nivel de brillo correspondiente al color azul.

Si el LED WS2812B deja de recibir datos por un tiempo mayor o igual a 50 microsegundos, entonces el circuito de control se reinicia y se prepara para recibir nuevos datos de control.

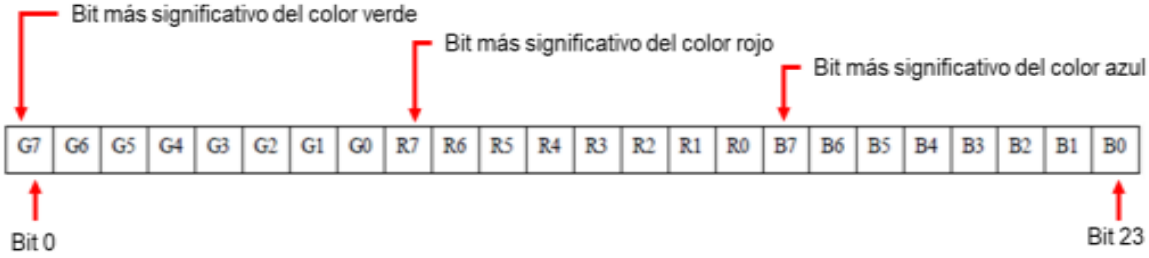


Figura 16. Composición de datos de 24 bits.
Fuente: (Worldsemi, 2023).

4.4.2 Procesador de video

El procesador de video LED es un equipo de control y procesamiento de imágenes de alto rendimiento para pantallas LED a todo color. Basado en el procesamiento de imágenes de video y la tecnología de procesamiento de señales de alta definición. Se adoptan el diseño de hardware patentado y los requisitos especiales de la pantalla LED a todo color. Puede recibir y procesar una variedad de señales de gráficos de video diferentes al mismo tiempo y mostrarlas en el LED a todo color (LED, 2023).

La tarea del procesador de video que debe completar es transformar la señal de imagen desde el exterior. (como Blu-ray DVD, computadora, caja de reproductor HD, etc.) en la señal que la pantalla LED puede aceptar, y optimizar la imagen de video para que la pantalla LED se muestre más con mejor resolución.

4.4.3 Controladora de pantalla de LEDs

Los LED WS2812B no se iluminan solos; requieren un microcontrolador y programación. Existen múltiples opciones a considerar para el prototipo, los cuales se muestran en la **Figura 17**.

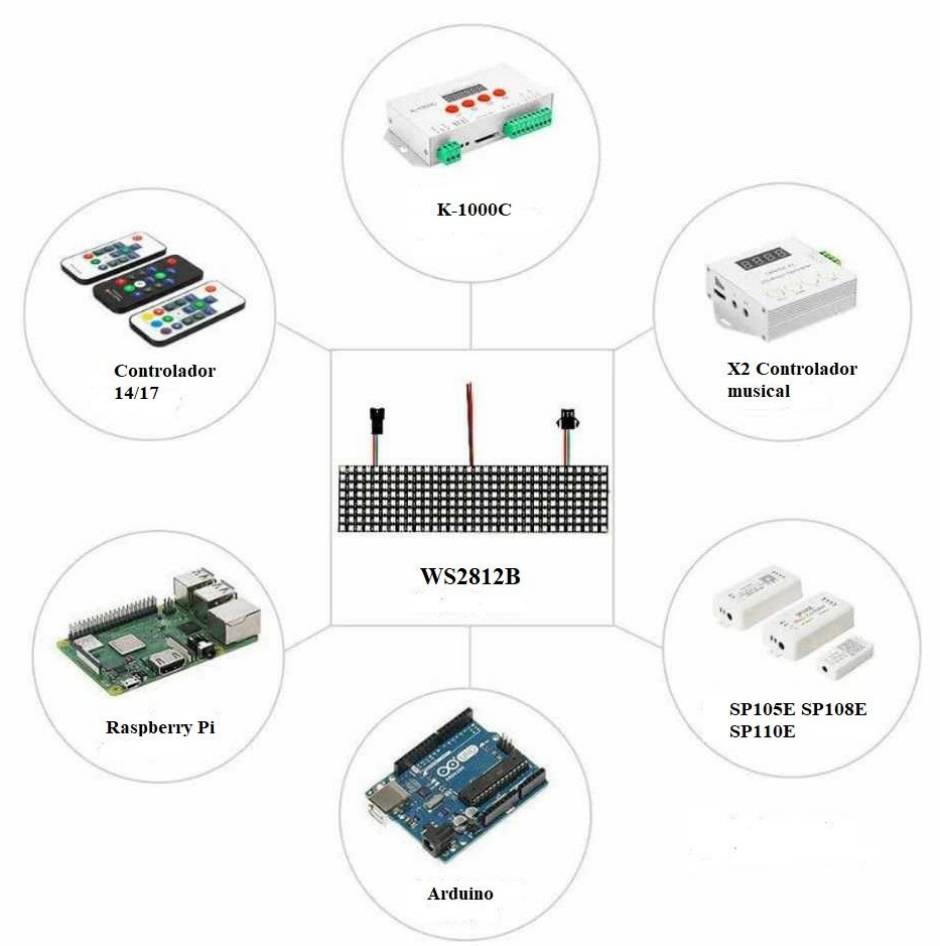


Figura 17. Controladores para LEDs.
Fuente: (Alibaba, 2023).

No existe un límite inherente en la longitud máxima de una cadena de LEDs, pero eventualmente se debe considerar varios límites que se describen a continuación:

- RAM: Los LED WS2812B requieren algo de RAM del microcontrolador host; más píxeles equivale a más RAM. Son solo unos pocos bytes cada uno, pero como la mayoría de los microcontroladores tienen recursos bastante limitados, esto se convierte en una limitante para pantallas de dimensiones considerables.
- Energía: Cada LED consume muy poca corriente; más píxeles igual a más potencia requerida. Las fuentes de alimentación también tienen un límite superior.
- Tiempo: Los WS2812B procesan datos del microcontrolador host a una velocidad de datos fija; más píxeles igual a más tiempo y velocidades de fotogramas de animación más bajas.

Tomando en consideración dichas limitaciones las mejores opciones para un prototipo de controlador de pantalla LED son la RASPBERRY PI, ARDUINO DUE y el ESP32.

4.4.3.1 Raspberry Pi

Es un computador de placa reducido o placa única (SBC-Single Board Computer) de bajo coste y con un tamaño compacto, pero ofrece unas prestaciones muy potentes en cuanto a otras placas de desarrollo, un modelo de esta placa se puede observar en la **Figura 18**.

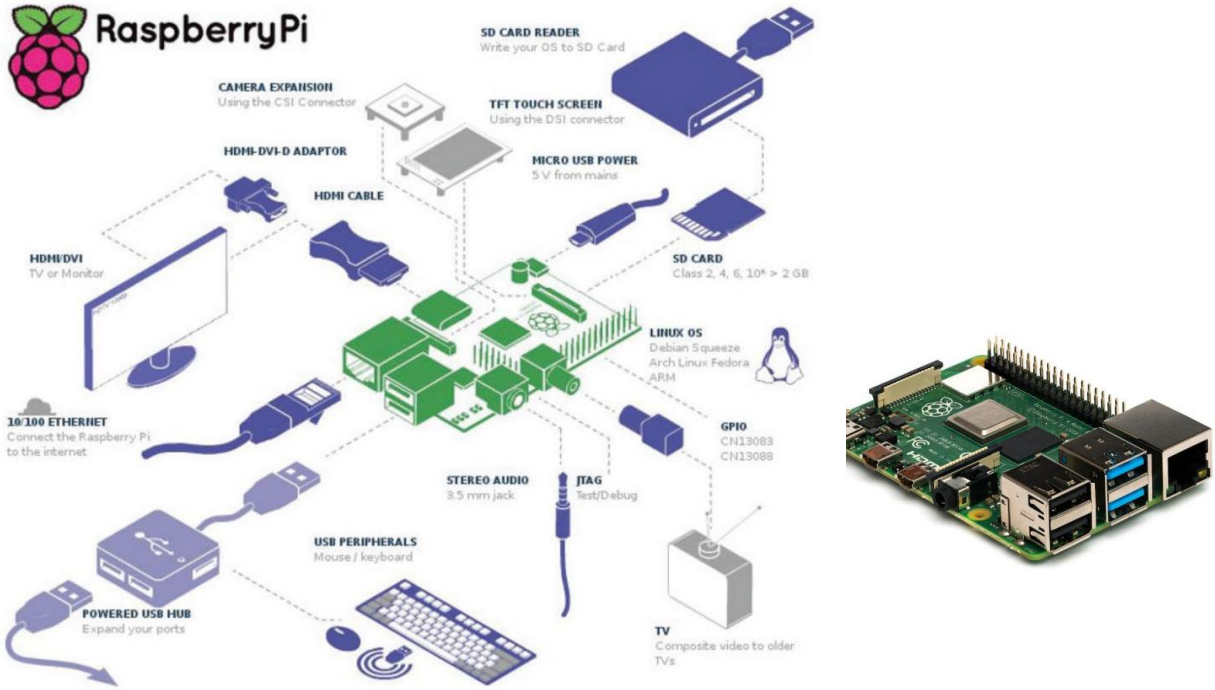


Figura 18. a) Resumen de las conexiones hardware en Raspberry Pi. b) Imagen de la placa Raspberry Pi 4B.
Fuente: a) Obtenido de (https://www.steren.com.ec/raspberry-pi4-b.html) b) *(López, 2017)*.

4.4.3.2 Arduino DUE

Arduino es una plataforma de hardware de código abierto ampliamente utilizada en proyectos de electrónica y robótica. El Arduino Due es uno de los modelos más potentes de las placas Arduino (**Figura 19**), esta basado en un potente microcontrolador SAM3X8E ARM Cortex-M3 que incorpora todas las funcionalidades clásicas de Arduino y añade otra nuevas.

Ofrece un total de 54 pines de entrada/salida (12 de las cuales son PWM con resolución configurable), 12 entradas analógicas con una resolución de 12 bits, 4 puertos UART por hardware y dos conversores DAC (digital a analógico), un resonador de cuarzo de 84MHz, dos conexiones USB (una de programación y otra que puede actuar como USB Host).

Figura 19. Placa Arduino DUE.
Fuente: (ARDUINO, 2023).

También incluye los pines de programación ICSP y JTAG, el voltaje máximo de los pines es de 3,3 V por lo que hay que tener precaución y no conectar dispositivos de 5 V ya que podrían dañar la placa. Arduino se ha convertido en una opción de bajo coste para el control de matrices LED RGB, gracias a su facilidad de programación y su amplia comunidad de usuarios, Arduino proporciona una solución flexible y accesible para la creación y el control de pantallas informativas basadas en matrices LED RGB. La combinación de Arduino con las matrices WS2812B ofrece un entorno de desarrollo accesible para la implementación de diversos proyectos en áreas como la señalización digital, la visualización de datos y el arte interactivo.

Las principales características de esta placa de hardware se pueden observar en la **Tabla 4**. Las características completas se exponen en el **Anexo** .

Tabla 4. Características Arduino DUE.

Especificaciones técnicas	
Tensión de funcionamiento	3,3V
Voltaje de entrada	7-12V
Memoria flash	512 KB
SRAM	96KB
Velocidad de reloj	84MHz

Fuente: (ARDUINO, 2023).

4.4.3.3 ESP32

Es una placa de desarrollo de hardware con funciones que incluyen Wi-Fi y conectividad Bluetooth. En la **Figura 20** se puede observar un modelo de esta placa.

Figura 20. Placa ESP32.
Fuente: (ESPRESSIF, 2023).

Las principales características sobre esta placa se pueden observar en la **Tabla 5**, las características completas se exponen en el **Anexo** .

Tabla 5. Características ESP32.

Especificaciones técnicas	
Tensión de funcionamiento	3,3V
Voltaje de entrada	3-3,6V
Memoria flash	448 KB
SRAM	520 KB
Velocidad de reloj	240MHz

Fuente: (ESPRESSIF, 2023).

4.4.4 Fuente de alimentación

Para garantizar el buen funcionamiento del circuito de varias matrices de LEDs, es importante contar con un suministro de energía constante y proporcional al tamaño de la pantalla, y eso se determina de acuerdo a la cantidad de matrices LED a utilizar.

Romero (2013), indica que para realizar un trabajo en lo que respecta a los tableros de tecnología LED, para obtener una buena intensidad luminosa, es necesario que se apoyen correctamente los LED a la energía que fluye por ellos, y así de esta manera poder evitar que se puedan deteriorar; sin embargo, hay que considerar que el voltaje va a ir de acuerdo al tipo y modelo de LED a utilizar.

Para determinar la fuente de alimentación para una pantalla informativa implica considerar varios factores, como la cantidad de LEDs, su consumo de energía y la tensión requerida. El consumo eléctrico de cada LED se determinará mediante las especificaciones técnicas del modelo de matrices elegidas. Para obtener el consumo de energía total de la pantalla se debe considerar el consumo de energía de un solo LED (en amperios) por el número total de LEDs en la pantalla.

4.4.4.1 Dimensionamiento de cables

Para el cálculo de la sección de cable en los distintos tramos o derivaciones donde circula la corriente continua (directa) se realiza a partir de la ecuación 1:

$$S = \frac{1.5 * L * I}{\Delta V * C} \qquad \text{Ecuación 1}$$

Donde:

S : Sección del cable conductor en $[mm^2]$.

L : Longitud del cable conductor en ese tramo en $[m]$.

I : Intensidad de corriente máxima que circula por el conductor en $[A]$.

ΔV : Es la caída de tensión máxima permitida en los conductores, que según se indica en el Pliego de Condiciones Técnicas del IDAE, deberá ser en los conductores de continua como máximo del 3% $[V]$.

C : Conductividad del material que forma el conductor, en este caso cobre, cuya conductividad a 20℃ es de 56 $\left[\frac{m}{\Omega * mm^2}\right]$.

En la **Tabla 6** se indica la capacidad de corriente para determinar la sección del cable necesaria en función de la corriente total y la longitud del cable.

Tabla 6. Sección de cable y su capacidad de corriente.

Sección de cable (mm2)	Intensidad Máxima (A)
0,5	6
0,75	9
1	11
1,5	14
2	16
2,5	20
3,5	25
4	28
6	37
8	48
10	53
16	75
25	100
35	125
50	160

Fuente: *(COELECTRIX Corporation, 2024)*.

4.4.4.2 Librerías

Algunas de las principales bibliotecas de Arduino que se utilizan para el control de matrices LED RGB, especialmente las matrices WS2812B, se describen a continuación:

4.4.4.2.1 Adafruit NeoPixel

La biblioteca Adafruit NeoPixel para Arduino es una herramienta fundamental en el campo de la iluminación controlada por microcontroladores. Desarrollada por Adafruit Industries, esta biblioteca proporciona una interfaz fácil de usar para controlar matrices y tiras de LEDs direccionables, como LEDs NeoPixel (WS2812, WS2811, SK6812) y DotStar (Adafruit Industries, 2024).

- ***Funcionalidades Principales:***

Control Preciso de LEDs: La biblioteca permite controlar cada LED individualmente, lo que permite una amplia gama de efectos de iluminación, desde animaciones simples hasta patrones complejos.

Compatibilidad: Es compatible con una variedad de dispositivos basados en LEDs, incluyendo matrices, tiras, anillos y matrices de matriz.

Facilidad de Uso: La interfaz de la biblioteca se ha diseñado para ser intuitiva y fácil de entender.

Personalización: Ofrece una amplia gama de funciones y efectos predefinidos, así como la capacidad de crear efectos personalizados y ajustar parámetros como el brillo, la velocidad y la dirección de desplazamiento.

Optimización de Memoria: Está optimizada para minimizar el uso de memoria RAM y de programa, lo que la hace adecuada incluso para proyectos con recursos limitados.

- ***Uso y Aplicaciones:***

Iluminación Decorativa: Es ideal para proyectos de iluminación decorativa, como luces de ambiente, iluminación de fiestas y exhibiciones artísticas.

Proyectos de Vestuario: Se utiliza comúnmente en proyectos de vestuario, permitiendo la creación de prendas de vestir y accesorios iluminados y personalizados.

Señalización y Visualización: Se puede utilizar en proyectos de señalización y visualización, como carteles luminosos, marcadores de tiempo y sistemas de información.

Electrónica de Entretenimiento: Es perfecta para proyectos de electrónica de entretenimiento, como juegos de luces interactivos, instrumentos musicales luminosos y dispositivos de control de luces para eventos.

Requerimientos de Potencia: Las matrices y tiras de LEDs pueden requerir una cantidad significativa de potencia, especialmente cuando se utilizan muchos LEDs a la vez. Es importante calcular y proporcionar la potencia adecuada para evitar problemas de rendimiento y daños en los LEDs.

Configuración Correcta: Es importante configurar correctamente los pines de datos, el tipo de LEDs y otros parámetros en el código para garantizar un funcionamiento adecuado de la biblioteca.

4.4.4.2.2 FastLED

La biblioteca FastLED es una herramienta de software de código abierto diseñada para el control avanzado de matrices de LEDs y tiras de LEDs direccionables en Arduino y otras plataformas compatibles, ofrece un alto rendimiento y una amplia gama de funciones para la creación de efectos de iluminación dinámicos y personalizados (Kriegsman & Garcia, 2013).

Soporte para una Amplia Gama de LEDs: FastLED es compatible con una variedad de tipos de LEDs, entre los que se destacan WS2812, WS2811, WS2801, APA102, entre otros. Esto permite a los usuarios crear proyectos con una amplia variedad de opciones de iluminación, desde tiras individuales hasta matrices complejas (GitHub Repository, 2024).

Control Avanzado de Color: La biblioteca ofrece un control preciso sobre el color de cada LED, lo que permite la creación de efectos visuales y animaciones personalizadas. Los usuarios pueden especificar los valores RGB de cada LED individualmente o utilizar funciones predefinidas para generar patrones de color dinámicos.

Animaciones Predefinidas: FastLED incluye una variedad de efectos visuales predefinidos, como arco iris, fuego, auroras boreales y muchos más. Estas animaciones pueden ser fácilmente integradas en proyectos sin necesidad de programación adicional, lo que permite a los usuarios crear efectos visuales.

Optimización de Rendimiento: La biblioteca ha sido cuidadosamente optimizada para maximizar el rendimiento en plataformas Arduino, lo que permite una reproducción sin interrupciones de animaciones incluso en proyectos con grandes cantidades de LEDs. Esto se logra a través de técnicas avanzadas de programación y gestión eficiente de la memoria.

Facilidad de Uso: FastLED está diseñada para ser fácil de usar, con una interfaz intuitiva que permite a los usuarios comenzar a crear proyectos rápidamente. La biblioteca viene con una documentación detallada y ejemplos de código que cubren una amplia gama de aplicaciones y escenarios de uso.

4.4.4.2.3 NeoMatrix

La biblioteca Adafruit NeoMatrix es una herramienta poderosa y versátil para el control de matrices de LEDs RGB, especialmente diseñada para trabajar en conjunto con la biblioteca Adafruit NeoPixel. Esta biblioteca proporciona una interfaz fácil de usar para crear y controlar matrices de LEDs de forma eficiente y efectiva en proyectos de electrónica y programación con Arduino.

Una de las características más destacadas de la biblioteca Adafruit NeoMatrix es su capacidad para controlar matrices de LEDs de cualquier tamaño, desde pequeñas matrices 8x8 cm hasta grandes pantallas de LEDs de alta resolución. Esto permite a los usuarios crear una amplia variedad de proyectos, como carteles de mensajes, relojes, gráficos animados, y más, utilizando una sola biblioteca.

La biblioteca Adafruit NeoMatrix ofrece una serie de funciones y herramientas que simplifican el proceso de programación y control de matrices de LEDs. Esto incluye funciones para dibujar formas geométricas simples, como líneas, círculos y rectángulos, así como para escribir texto y mostrar imágenes en la matriz de LEDs. Además, la biblioteca proporciona soporte para animaciones personalizadas, lo que permite a los usuarios crear efectos visuales dinámicos y atractivos con facilidad.

Otra característica importante de la biblioteca Adafruit NeoMatrix es su capacidad para gestionar la multiplexación de matrices de LEDs, lo que permite controlar múltiples matrices con un número limitado de pines de salida en Arduino. Esto es especialmente útil en proyectos que requieren la construcción de grandes pantallas de LEDs utilizando múltiples matrices.

4.4.4.2.4 Adafruit GFX

La biblioteca Adafruit GFX es una herramienta esencial y versátil para cualquier desarrollador que trabaje con dispositivos y pantallas gráficas en proyectos de Arduino. Esta biblioteca proporciona una amplia gama de funciones y herramientas que simplifican el proceso de creación de interfaces de usuario gráficas y la visualización de información en pantallas (Adafruit Industries (s.f.), 2024).

Una de las características más destacadas de la biblioteca Adafruit GFX es su capacidad para abstraer la complejidad de los dispositivos gráficos, permitiendo a los usuarios trabajar con una variedad de pantallas y controladores de forma sencilla y uniforme. Esto significa que los desarrolladores pueden escribir código una vez y ejecutarlo en diferentes tipos de pantallas, sin tener que preocuparse por los detalles específicos de cada dispositivo.

La biblioteca Adafruit GFX proporciona una serie de funciones para dibujar formas geométricas simples, como líneas, círculos, rectángulos y polígonos, así como para escribir

texto en diferentes fuentes y tamaños. Esto permite a los usuarios crear interfaces de usuario gráficas personalizadas y mostrar información de manera clara y legible en las pantallas.

Además de las funciones básicas de dibujo, la biblioteca Adafruit GFX ofrece soporte para la creación de imágenes y gráficos personalizados, lo que permite a los usuarios importar y mostrar imágenes en formatos comunes como BMP y JPEG en las pantallas. Esto es especialmente útil en proyectos que requieren la visualización de gráficos complejos o la incorporación de imágenes en la interfaz de usuario.

Otra característica importante de la biblioteca Adafruit GFX es su capacidad para gestionar la memoria de pantalla de forma eficiente, lo que permite a los usuarios trabajar con pantallas de diferentes tamaños y resoluciones sin sacrificar el rendimiento o la estabilidad del sistema. Esto es especialmente importante en proyectos con recursos limitados de memoria, donde cada byte cuenta.

4.4.4.2.5 TFT_eSPI

La biblioteca TFT_eSPI es una herramienta fundamental en el desarrollo de proyectos con pantallas TFT (Thin-Film Transistor) para microcontroladores como Arduino. Esta biblioteca, ofrece una interfaz fácil de usar y potentes funcionalidades para el control de pantallas TFT en proyectos de electrónica y programación (Bodmer, 2024).

Una de las características más destacadas de TFT_eSPI es su amplia compatibilidad con una gran variedad de controladores de pantalla TFT, incluyendo controladores ILI9341, ST7735, y otros modelos comunes. Esta versatilidad permite a los usuarios utilizar la misma biblioteca para una variedad de pantallas TFT de diferentes tamaños y resoluciones, simplificando el proceso de desarrollo y facilitando la reutilización de código en diferentes proyectos.

La biblioteca TFT_eSPI ofrece una serie de funciones para realizar operaciones gráficas avanzadas en pantallas TFT, incluyendo el dibujo de formas geométricas simples (como líneas, círculos, rectángulos y polígonos), la escritura de texto en diferentes fuentes y tamaños, y la visualización de imágenes en formatos comunes como BMP y JPEG. Esto permite a los usuarios crear interfaces de usuario gráficas personalizadas y mostrar información de manera clara y legible en las pantallas.

Además de las funciones básicas de dibujo, TFT_eSPI proporciona soporte para características avanzadas como el desplazamiento de pantalla, el control de brillo y contraste, y la gestión de eventos táctiles en pantallas táctiles capacitivas y resistivas. Esto permite a los usuarios crear interfaces de usuario interactivas y aplicaciones multimedia avanzadas con facilidad.

Otra característica destacada de TFT_eSPI es su eficiente gestión de la memoria de pantalla, que permite a los usuarios trabajar con pantallas de diferentes tamaños y resoluciones sin sacrificar el rendimiento o la estabilidad del sistema. Esto es especialmente importante en proyectos con recursos limitados de memoria, donde cada byte cuenta.

4.4.4.3 Image Converter (UTFT)

La herramienta Image Converter UTFT es un recurso valioso y eficaz en el ámbito del desarrollo de proyectos con pantallas TFT (Thin-Film Transistor), especialmente cuando se utiliza en conjunto con la biblioteca UTFT para Arduino. Esta herramienta, diseñada para simplificar el proceso de conversión de imágenes para su uso en proyectos de electrónica y programación, ofrece una serie de funcionalidades que facilitan la integración de gráficos y contenido visual en pantallas TFT.

Una de las características más destacadas de Image Converter UTFT es su capacidad para convertir imágenes en una variedad de formatos comunes, como BMP, JPEG y PNG, en formatos compatibles con la biblioteca UTFT. Esto permite a los usuarios importar y mostrar imágenes de manera sencilla y eficiente en pantallas TFT, sin necesidad de realizar conversiones manuales o tediosas.

La herramienta Image Converter UTFT ofrece una interfaz fácil de usar que permite a los usuarios seleccionar rápidamente las imágenes que desean convertir, así como configurar opciones como el tamaño de la imagen, la resolución y el formato de salida. Una vez configurada, la herramienta realiza automáticamente la conversión y genera el código necesario para mostrar la imagen en una pantalla TFT utilizando la biblioteca UTFT.

Otra característica destacada de Image Converter UTFT es su capacidad para optimizar el uso de memoria en proyectos con recursos limitados, como Arduino. La herramienta genera código optimizado que minimiza el uso de memoria RAM y PROGMEM, lo que permite a los usuarios trabajar con imágenes de gran tamaño y resolución sin condicionar el rendimiento o la estabilidad del sistema.

5. Metodología

5.1 Equipos y materiales

Los equipos y materiales empleados para el desarrollo del proyecto son los siguientes

5.1.1 Recursos Tecnológicos

- **Softwares**
  - Word, Excel
  - AutoCAD 2024
- **Equipos**
  - Computador portátil.

5.1.2 Recursos Humanos

- Director de tesis
- Estudiante

5.1.3 Materiales

5.1.3.1 Materiales de construcción.

En la **Tabla 7** se detallan los equipos y materiales seleccionados para la construcción del prototipo.

Tabla 7. Materiales de construcción.

Cantidad	Descripción
12	Matriz LED flexible RGB WS2812B 16x16.
1	Fuente de alimentación JPS300V-5V 180A 900W.
1	Arduino DUE.
1	ESP32/ESP-WROOM-32S.
1	Display OLED I2C 0.96inch 64x128.
1	Pulsador decodificador rotatorio de 360 grados EC11.
1	Adaptador ESP32.
1	Diodo rectificador 800V 1N4006.
1	Módulo de memoria para Micro SD Card.
1	Micro SD 8GB.
1	6 m de cable de cobre #12

5.1.3.2 Materiales adicionales.

En la **Tabla 8** se muestran los materiales adicionales para el prototipo de pantalla informativa.

Tabla 8. Materiales adicionales.

Cantidad	Descripción
1	Marco de madera.
1/32	Pintura negra esmalte.
1	Caja de tornillos Drywall Screws.
1	Envoltura de vinilo de fibra de carbono.

5.2 Procedimiento

Para el desarrollo y cumplimiento de los objetivos planteados en el presente proyecto, se estructuró y desarrolló la siguiente metodología.

5.2.1 Metodología para cumplimiento del primer objetivo especifico

Para la ejecución del primer objetivo que es: "Analizar tecnologías LED RGB para la construcción de pantallas informativas con matrices LED", se efectuó una revisión bibliográfica de literatura técnica de las generalidades de tecnologías LED RGB, características de controladores y los inconvenientes que presentan, en la **Figura 21** se muestra el procedimiento ejecutado.

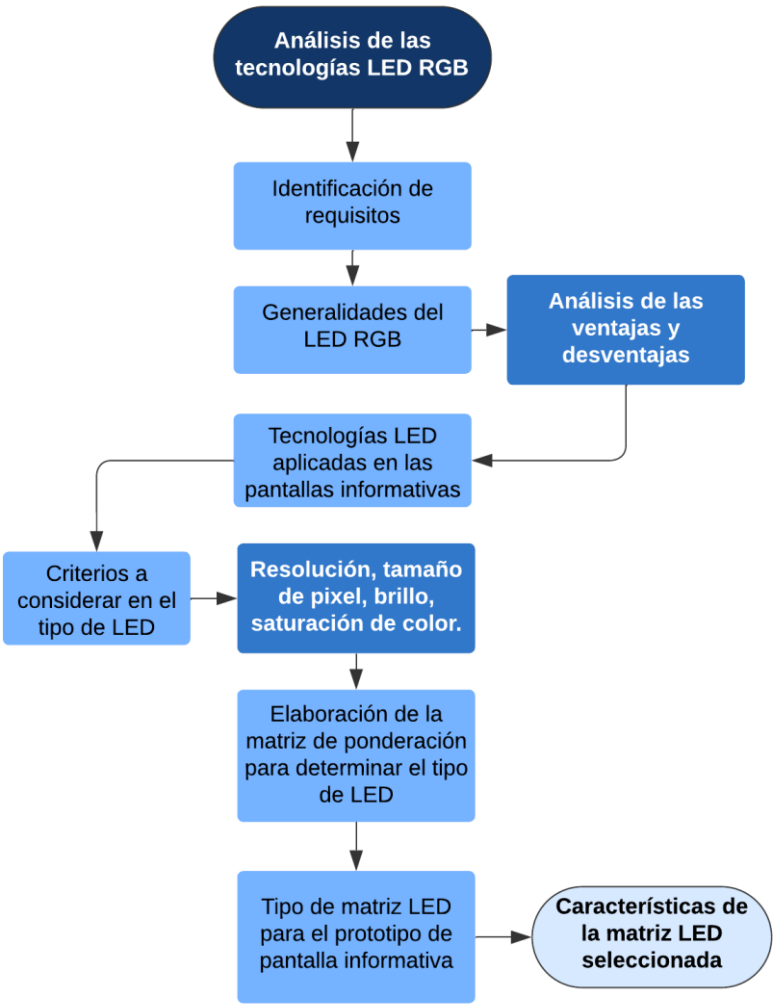


Figura 21. Metodología aplicada para el desarrollo del primer objetivo.

Revisión de literatura y análisis del estado del arte:

- Búsqueda sistemática de bases de datos académicas y recursos en línea para identificar estudios, artículos y documentos relevantes sobre tecnologías LED RGB y su aplicación en pantallas informativas y el uso de Arduino y ESP32 como controlador.
- Recopilar información de características técnicas, ventajas, desventajas y aplicaciones prácticas de diferentes tecnologías LED RGB disponibles en el mercado, efectuar un análisis de trabajos previos y proyectos similares para identificar enfoques, técnicas y metodologías utilizadas.

Evaluación de características técnicas:

- Analizar las especificaciones técnicas de las tecnologías LED RGB, incluyendo la resolución, el tamaño de píxel, el brillo, la saturación de color y la uniformidad de la iluminación.
- Comparar y contrastar las características técnicas de diferentes tecnologías LED RGB para identificar las que se adapten a los requisitos de la pantalla informativa del proyecto.

Análisis de costos y viabilidad:

- Evaluar los costos que representa la implementación del prototipo de pantalla LED RGB, con materiales disponibles en el mercado nacional e internacional.

Síntesis de resultados:

- Integrar todos los criterios identificados en la revisión de literatura, la evaluación de características técnicas, la investigación de casos de estudio y el análisis de costos para identificar los criterios que rigen el uso de las tecnologías LED RGB para pantallas informativas.

5.2.2 Metodología para cumplimiento del segundo objetivo especifico

Para el desarrollo del segundo objetivo: “Dimensionar componentes y materiales para la pantalla informativa escalable con tecnologías LED RGB”, se utiliza las recomendaciones expuestas en la literatura técnica referente a tecnologías LED RGB, se identifica los materiales necesarios para la construcción de la pantalla, la determinación del tipo de matriz LED a utilizarse con el hardware de control, ya que de esto depende la selección del resto de equipos a utilizarse en el prototipo, en la **Figura 22**, se exponen las actividades previas al diseño y construcción del prototipo.

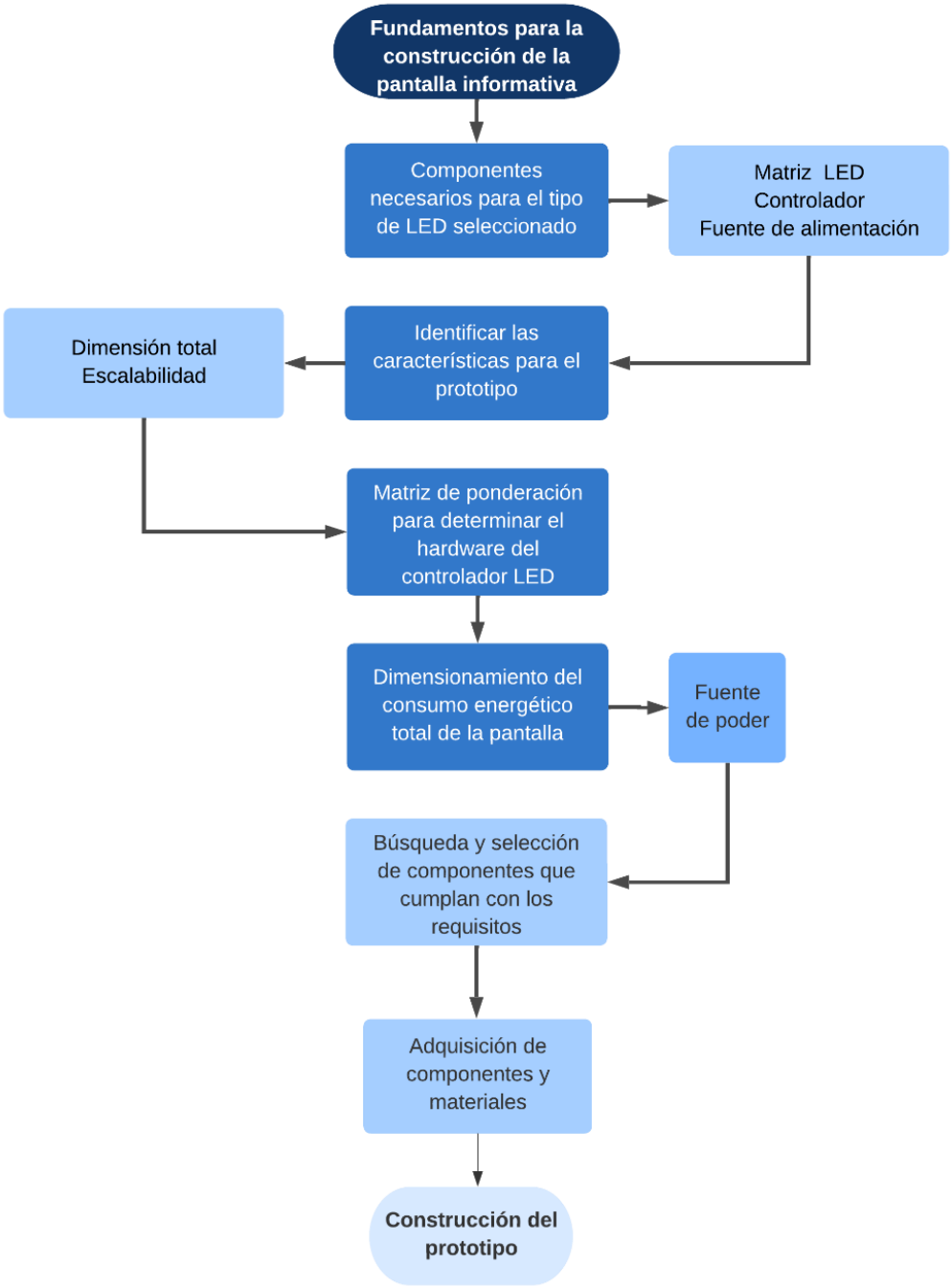


Figura 22. Metodología aplicada para el desarrollo del segundo objetivo.

Identificación de requisitos técnicos:

- Identificar requisitos técnicos específicos para el prototipo de pantalla informativa considerando resolución, dimensiones, luminosidad requerida.
- Establecer criterios de selección para los componentes y materiales en función de los requisitos técnicos identificados y las características de las tecnologías LED RGB disponibles en el mercado.

Investigación de opciones de componentes:

- Evaluar diferentes opciones de componentes y materiales necesarios para la construcción del prototipo de pantalla informativa, incluyendo matrices LED RGB, controladores Arduino, fuentes de alimentación, estructuras de montaje.

Evaluación de compatibilidad:

- Analizar la compatibilidad de los diferentes componentes y materiales seleccionados para garantizar la integración tecnológica de los diferentes componentes del prototipo de pantalla informativa.
- Considerar la interoperabilidad, la capacidad de expansión y la escalabilidad del sistema para adaptarse a futuras actualizaciones y mejoras del prototipo.

Cálculo de requerimientos de potencia y alimentación:

- Determinar los requerimientos eléctricos para el suministro eléctrico del prototipo de pantalla informativa en función de la cantidad de matrices LED RGB, y otros componentes electrónicos.
- Realizar un análisis para dimensionar la capacidad de la fuente de alimentación y evitar sobrecargas o fallos en el sistema.

Diseño de la estructura y montaje:

- Diseñar la estructura física de la pantalla informativa considerando la disposición de las matrices LED RGB, los requisitos de ventilación y acceso a los componentes.
- Procedimiento de ensamblaje adecuado para garantizar el funcionamiento del prototipo de pantalla informativa.

Estimación de costos y presupuesto:

- Estimar los costos asociados con la adquisición de componentes y materiales, la fabricación de la estructura y el montaje del prototipo de pantalla informativa.
- Elaborar un análisis de precios considerando todos los componentes que conforman el prototipo de pantalla informativa.

- **Programación de eventos en la pantalla informativa**

La programación de eventos en la pantalla informativa se refiere a la capacidad de definir acciones o cambios en la visualización de datos en respuesta a ciertas condiciones o eventos específicos. Esto puede incluir desde la activación de efectos visuales en función del tiempo hasta la actualización de la información mostrada en respuesta a datos externos.

Temporizadores y secuencias temporales:

- Implementación de temporizadores para activar eventos en momentos específicos del día o en intervalos regulares.
- Creación de secuencias temporales para la reproducción de animaciones o cambios graduales en la visualización de datos a lo largo del tiempo.

Detección de eventos externos:

- Desarrollo de rutinas de detección de eventos externos, como la recepción de datos e información de fuentes externas (PC).
- Utilización de interrupciones o bucles de espera activa para detectar cambios en los datos de entrada y activar acciones correspondientes en la pantalla informativa.

Interacción con el usuario:

- Incorporación de eventos de interacción con el usuario, como la pulsación de botones o la detección de gestos en una pantalla táctil.
- Definición de acciones específicas en respuesta a las interacciones del usuario, como cambiar el modo de visualización o actualizar la información mostrada.

Integración con sistemas externos:

- Establecimiento de conexiones con sistemas externos, como lecturas de memorias externas SD, para recibir información actualizada de manera periódica o en tiempo real.
- Configuración de eventos de sincronización para actualizar la visualización de datos en función de la información recibida de sistemas externos.

Pruebas y ajustes:

- Realización de pruebas exhaustivas para validar el funcionamiento de los eventos programados en la pantalla informativa en diferentes escenarios y condiciones.
- Ajuste de parámetros y configuraciones en función de los resultados obtenidos en las pruebas, con el fin de optimizar la eficiencia y la efectividad de los eventos programados.

5.2.3 Metodología para cumplimiento del tercer objetivo

Para cumplir el tercer objetivo del proyecto que es: “Validar el prototipo de pantalla informativa utilizando matrices LED RGB para proyectar texto alfanumérico e imágenes”, se dispone la metodología señalada en la **Figura 23**, donde se detalla las actividades ejecutadas para la demostración de rendimiento y aplicación del prototipo construido.

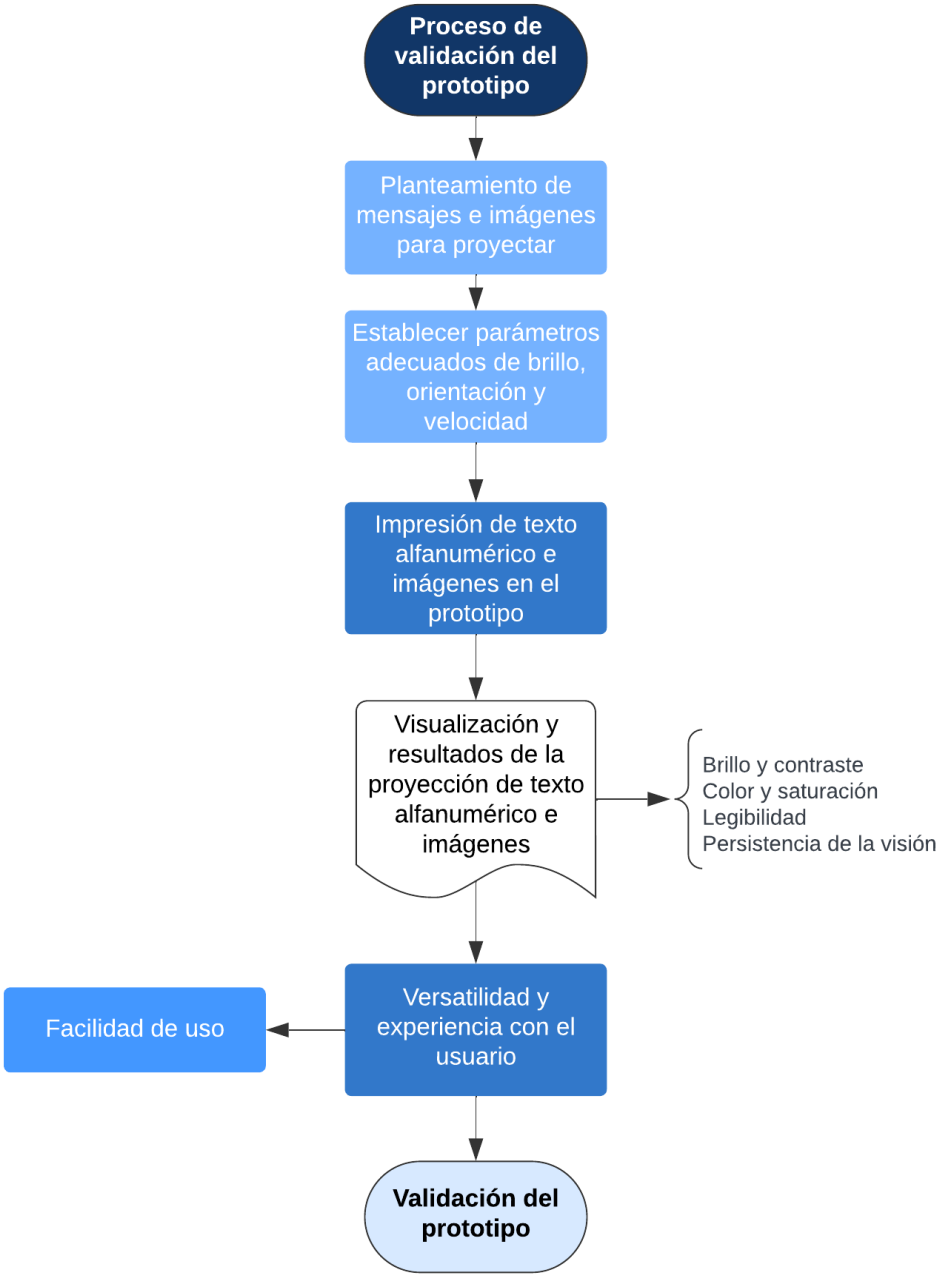


Figura 23. Metodología aplicada para el desarrollo del tercer objetivo.

Definición de criterios de validación:

- Establecer criterios para la evaluación del prototipo de pantalla informativa, considerando aspectos técnicos (resolución, brillo, contraste), funcionales (funcionamiento del controlador, visualización de contenido) y de utilidad (interfaz de usuario, experiencia del usuario).

Planificación de pruebas y experimentos:

- Definir escenarios de prueba que permitan identificar la fiabilidad del prototipo.

Ejecución de pruebas y experimentos:

- Realizar pruebas de funcionamiento para verificar el funcionamiento de las funcionalidades principales del prototipo, incluyendo la visualización de texto alfanumérico e imágenes, la respuesta a comandos de usuario y la estabilidad del sistema.

5.3 Diseño, dimensionamiento y construcción del prototipo

5.3.1 Criterio para la selección del tipo de matriz LED a implementarse

A partir del **Capítulo III** en donde se expone una comparación de los diferentes tipos de tecnologías para pantallas informativas, estableciendo brevemente las ventajas y desventajas de cada una, se elabora una matriz de ponderación, valores que se exponen en la **Tabla 9**, en donde se detallan los criterios más significativos para determinar la opción más adecuada entre los diferentes tipos de matrices LED existentes. El peso de cada criterio se define por consideraciones propias por parte del autor, consideraba de qué forma debe resaltar el prototipo.

Tabla 9. Matriz de ponderación para definir el tipo de LED a implementarse

Criterio	Peso del criterio	WS2801		Matriz de punto		Panel LED smd		WS2812B	
		Valor	%	Valor	%	Valor	%	Valor	%
Adaptabilidad en diferentes espacios	10%	5	10	4	8	5	10	5	10
Vida útil	12%	5	12	4	9,6	5	12	5	12
Calidad del color	15%	4	12	3	9	5	15	4	12
Brillo y luminosidad	10%	4	8	3	6	4	8	4	8
Eficiencia en el uso de recursos	8%	4	6	4	6,4	3	4,8	4	6,4
Costo	10%	3	6	4	8	2	4	5	10
Facilidad de control	17%	3	10	2	6,8	3	10,2	5	17
Durabilidad y flexibilidad	10%	5	10	3	6	3	6	5	10
Resolución y densidad de píxeles	5%	4	4	3	3	5	5	4	4
Diseño estético y de calidad	3%	4	2	3	1,8	5	3	4	2,4
TOTAL	100%		81		65		78		**92**

Estos valores reflejan una distribución de peso que asigna más importancia a las características fundamentales de la calidad visual y funcionalidad del dispositivo (como la calidad del color, el brillo y la resolución) además se considera aspectos prácticos como el costo y la facilidad de control. Donde el peso del criterio se refiriere a la importancia que cada uno representa respecto al resto de criterios, y la valoración (valor) se rige según la **Tabla 10**:

Tabla 10. Valoración aplicada a la matriz de ponderación.

Muy bajo	Bajo	Medio	Bueno	Muy bueno
1	2	3	4	5

Los diez criterios establecidos (**Tabla 9**) para la determinación del tipo de matriz LED RGB adecuada se basan en los requerimientos del prototipo de esta investigación, donde la facilidad de control se refiere a que el método y forma para hacer funcionar los LEDs no representen mayor complejidad; la calidad del color, es un indicador importante en este proyecto y se refiere a la eficacia de los colores que se pueda generar con la matriz y el costo. Estos son los criterios más importantes al momento de establecer el tipo de matriz a implementar en este prototipo.

5.3.2 Criterio para la selección del controlador.

En la actualidad existen varios hardware que pueden implementarse para el control de pantallas LED, cada uno cuenta con características propias basadas en la adaptabilidad, cantidad de recursos, velocidad y de más necesidades requeridas.

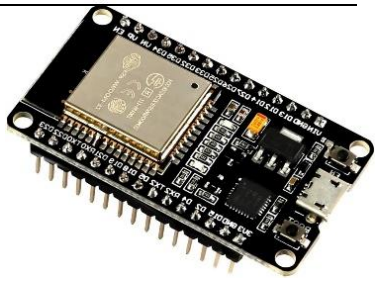
Para la selección del controlador más apropiado, que satisfaga los requerimientos de este proyecto, se consideran tres controladores, en la **Tabla 11** se observa las ventajas y desventajas consideradas para la elección del controlador.

Tabla 11. Ventajas y desventajas de hardware del controlador.

VENTAJAS	DESVENTAJAS	HARDWARE
• Potencia de procesamiento considerable en la versión 4. • Versatilidad para aplicarse en una amplia variedad de aplicaciones.	• Consumo de energía mayor a muchos microcontroladores. • Precio elevado para otros microcontroladores.	
• Óptimo para aplicaciones en tiempo real. • Entradas y salidas analógicas. • Compatibilidad con una gran gama de shields y módulos. • Precio accesible.	• Consumo de energía mayor al ser más potente que otros modelos de Arduino.	

- Diseño compacto.
- Potencia de procesamiento elevada por su doble núcleo.
- Bajo consumo de energía.

- Aprendizaje riguroso.
- Estabilidad del firmware.
- Compatibilidad del hardware.



Una vez detalladas las ventajas y desventajas de los tres dispositivos seleccionados, se observa que el dispositivo tres es el más adecuado basado en la potencia de procesamiento, ya que permite un control óptimo de las matrices, tiene un diseño compacto y su precio es accesible en comparación a los otros dos dispositivos en relación precio-potencia. Sin embargo, para facilitar el análisis de las características de interés que intervienen en el prototipo a construir se elabora una matriz de ponderación que se exponen en la **Tabla 12**, donde se establece una valoración según las prioridades del diseño y para cada uno de los dispositivos analizados.

Tabla 12. Matriz de ponderación para determinar el controlador a implementarse

Criterio	**Peso del criterio**	**WS2801**		**Matriz de punto**		**Panel LED smd**		**WS2812B**	
		Valor	%	Valor	%	Valor	%	Valor	%
Adaptabilidad en diferentes espacios	10%	5	10,00	4	8	5	10	5	10
Vida útil	12%	5	12,00	4	9,6	5	12	5	12
Calidad del color	15%	4	12,00	3	9	5	15	4	12
Brillo y luminosidad	10%	4	8,00	3	6	4	8	4	8
Eficiencia en el uso de recursos	8%	4	6,40	4	6,4	3	4,8	4	6,4
Costo	10%	3	6,00	4	8	2	4	5	10
Facilidad de control	17%	3	10,20	2	6,8	3	10,2	5	17
Durabilidad y flexibilidad	10%	5	10,00	3	6	3	6	5	10
Resolución y densidad de píxeles	5%	4	4,00	3	3	5	5	4	4
Diseño estético y de calidad	3%	4	2,40	3	1,8	5	3	4	2,4
TOTAL	100%		81,00		64,60		78,00		**91,80**

Cada uno de los ocho criterios establecidos en esta matriz cuentan con un peso o porcentaje según la importancia para el diseño a construirse, valorados según la **Tabla 10**, donde principalmente se busca potencia de procesamiento en contraste con la facilidad de programación y el costo de la placa de hardware. Esto al tratarse de un prototipo a escala para proyecciones futuras. Como se observa las placas de desarrollo Arduino Due y Esp32 tienen el mismo valor para las variables dadas y que son de relevancia para este proyecto.

5.3.3 Diseño del sistema de marco y soporte

Para determinar el diseño del marco de la pantalla se consideran las dimensiones de las matrices LED seleccionadas, de acuerdo a la **Tabla 1**, además por ser un dispositivo versátil utilizado en variedad de aplicaciones que incluyen pantallas informativas a proyectos de iluminación y arte interactivo (**Figura 10.a**). En total la pantalla estará conformada por 12 matrices LED RGB WS2812B de 16x16 cm, dando la dimensión total de la pantalla de 64x48 cm (**Figura 24).** Para mejorar la visualización de las matrices LEDs, se utiliza una hoja difusora colocada frente a las matrices.

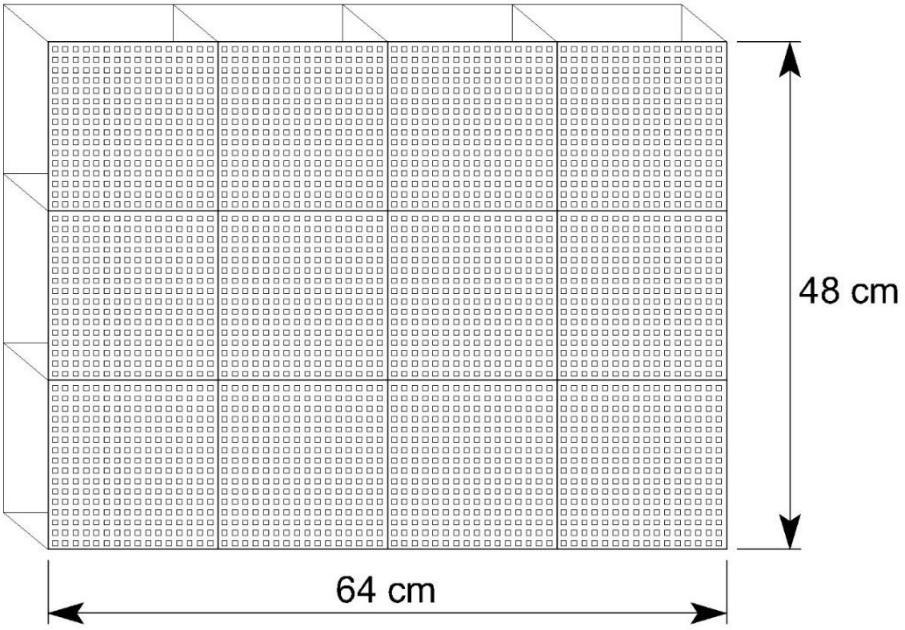


Figura 24. Dimensiones del prototipo de pantalla a implementarse.

Para determinar la versatilidad de los dispositivos de control de la pantalla se utiliza dos dispositivos, Arduino Due y ESP3-wrooom-32.

De forma horizontal se controlan los eventos de la pantalla con el controlador Arduino Due sin ningún módulo adicional, facilitando el uso de la pantalla; con el controlador ESP32 se configura la pantalla en formato vertical, al utilizar estas dos variantes de configuración de la pantalla se puede identificar la versatilidad de cada controlador (ver **Figura 25)**.

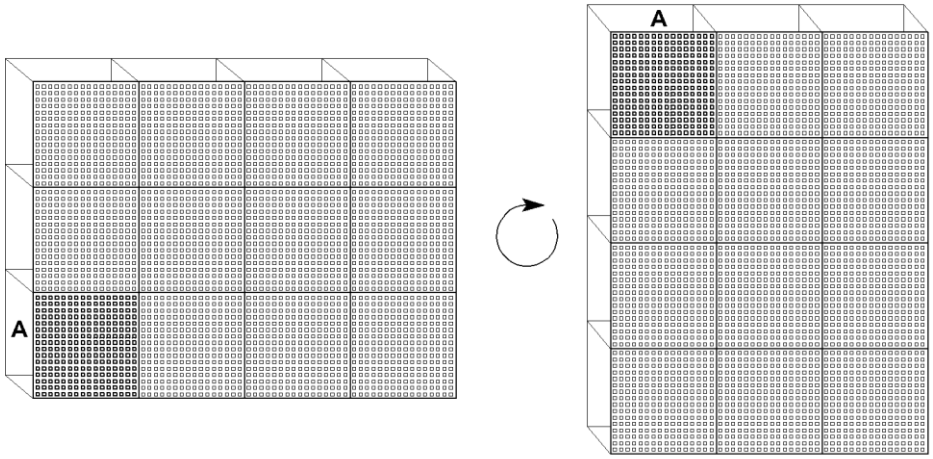


Figura 25. Formato horizontal y vertical del prototipo.

En el diseño y construcción se utilizaron cuatro perfiles de madera con sección de 2,5 cm x 2,5 cm, los cuales serán el soporte de todas las matrices, teniendo en consideración los formatos anteriores (horizontal y vertical), se consideró un sistema de eje central con un marco adicional posterior para el soporte general de la pantalla.

Se obtiene el marco con el sistema de giro para ambos formatos mediante el eje central, considerando la posición de los cables para que no interfiera con el movimiento de toda la pantalla, tanto del circuito de fuerza (energía) como de control (transferencia de datos).

5.3.4 Cálculo de la potencia necesaria y selección de la fuente de poder.

Para la selección de la fuente de poder los criterios de dimensionamiento se exponen a continuación.

5.3.4.1 Consumo por matriz LED WS2812B

Para el cálculo del consumo de cada matriz se emplea los datos de potencia de la **Tabla 1**, considerando un voltaje de funcionamiento de 5 V de las matrices LED, obteniendo el siguiente resultado:

$$I_{in} = \frac{0{,}3\ \frac{w}{pixel}}{5\ V} = 0{,}06\ \frac{A}{pixel} = 60\frac{mA}{pixel}$$

El número total de LEDs en cada matriz es de 256, por lo tanto, el consumo total por matriz es de:

$$I_m = 0{,}06\frac{A}{pixel} \times (256\ pixel)$$

$$I_m = 15{,}36\ A$$

5.3.4.2 Consumo total de la pantalla

Para determinar el consumo total de la pantalla se emplea la corriente por matriz y al tener un total de 12 matrices en toda la extensión de la pantalla, el consumo total es de:

$$I_T = 15{,}36A \times 12$$

$$I_T = 184{,}32\ A$$

Para seleccionar la fuente de poder se debe considerar la tensión de trabajo de los LED que es de 5 V y el consumo total de la pantalla que es de 184,32 A. Esto es un consumo muy elevado para una fuente de alimentación a 5 V. Los $60\ mA$ de cada LED es el consumo con el brillo máximo y en color blanco (la mezcla de los 3 colores RGB). Sin embargo, no siempre existe simultaneidad en que todos los píxeles estén activados. No se puede estimar un número único de LEDs activos para todos los eventos que se puedan reproducir, esto es aleatorio.

De acuerdo a adafruit (2023) en este tipo de LEDs se puede considera un consumo de $1/3$ del total ($20\ mA$ por píxel) como regla general sin tener efectos nocivos y sin perder la calidad de brillo en ciertos escenarios específicos. Por lo tanto, el consumo total de la pantalla se ajusta al siguiente valor:

$$I_m = 0{,}02\frac{A}{pixel}(256\ pixel) = 5{,}12\ A$$

$$I_T = 5{,}12\ A \times 12$$

$$I_T = 61{,}44\ A$$

Considerando una $I_T = 61{,}44\ A$, la fuente de alimentación seleccionada para el prototipo es la **JPS300V** con tres salidas de 5.0 V y 60 A, las características técnicas de este equipo se exponen en el **Anexo** .

5.3.5 Parámetros a controlar en la pantalla.

En la **Figura 26** se muestra un diagrama de los parámetros a controlar con las dos variantes de funcionamiento propuestas.

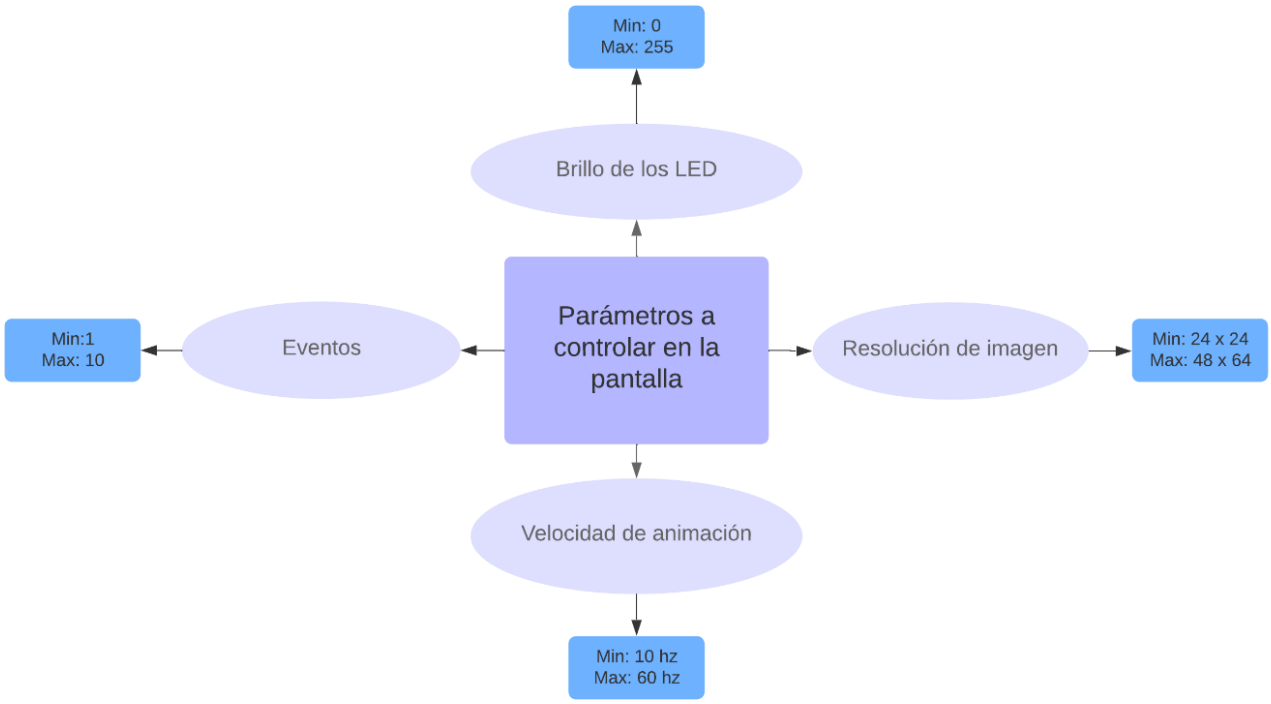


Figura 26. Parámetros a controlar en el prototipo de pantalla informativa.

5.3.6 Construcción del prototipo.

Una vez adquiridos los materiales establecidos en la **Tabla 7** y **Tabla 8**, se inicia la construcción del prototipo.

5.3.6.1 Red de alimentación de las matrices.

Para la realización de los orificios de conexión y distribución de energía es necesario elaborar una matriz de trazo o plantilla para asegurar la igualdad de distancias entre matrices en la plantilla de soporte. En la **Figura 27** se muestra parte del proceso en la plantilla de soporte de las matrices LED, *por* cada matriz se distribuyen tres orificios para energía y datos, para un total de 12 matrices de 16 x 16 cm para el prototipo.

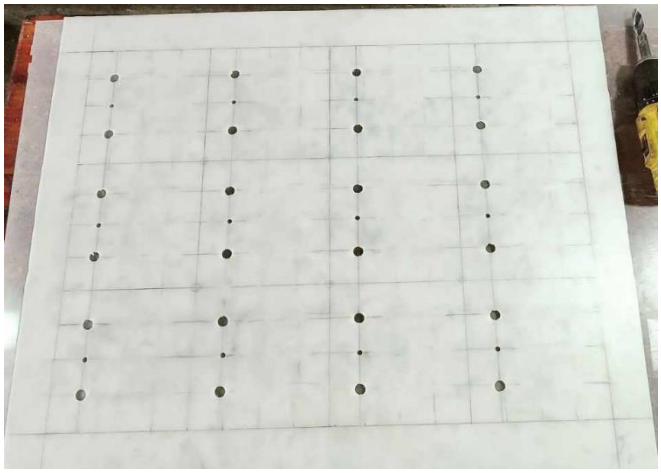


Figura 27. Distancias para transferencia de energía y datos en la pantalla.

Considerando que el consumo de todas las matrices es muy elevado y la fuente de alimentación tiene tres salidas DC, se utilizan tres derivaciones de conexión para alimentar a todas las matrices con 5VDC, cada derivación energiza a cuatro matrices, como se puede observar en la **Figura 28**.

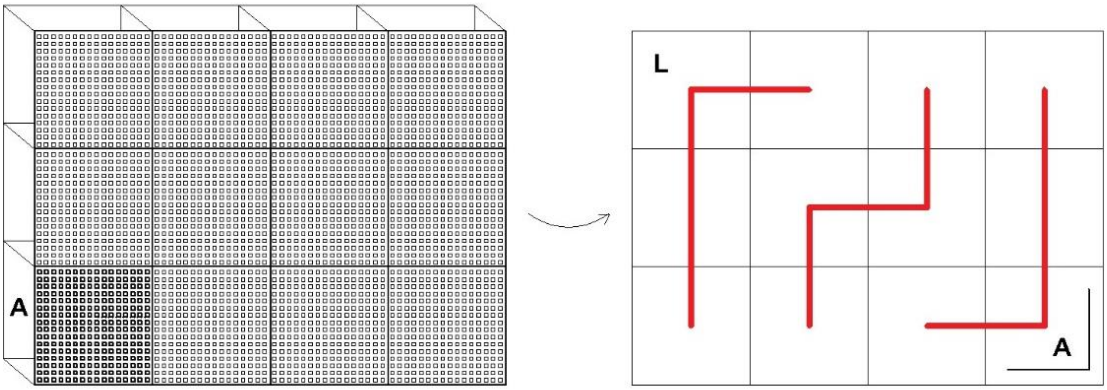


Figura 28. Rotación de la pantalla para observar la conexión de energía de las matrices (Vista posterior).

A continuación, todas estas derivaciones se encuentran en un solo punto de la parte posterior de soporte de la pantalla; sin que interfiera al momento de poder girar la pantalla a su formato vertical, se utiliza un cable adicional para el giro hasta llegar a la fuente de poder, ver **Figura 29**.

Figura 29. Sistema de energía; Distribución, salida de cables y colocación de fuente de poder.

5.3.6.1.1 Dimensionamiento de cables.

El cálculo de la sección del cable a utilizar en las matrices LED depende principalmente de la corriente de consumo del prototipo, el valor de diseño considerado es de 61.44 A, además de considerar la longitud de la conexión entre las matrices y la fuente de alimentación. Considerando el uso de las tres derivaciones de alimentación desde la fuente de poder, el consumo energético por derivación disminuye como se expone al aplicar el siguiente momento de cálculo:

$$I_s = \frac{61.44}{3} : \frac{A}{"derivaciones"}$$

$$I_s = 20.48A$$

La longitud del cable para cada derivación es de 1 m como máximo con una corriente de 20.48 A y 5 V. Aplicando la **Ecuación 1** para la sección del cable, se obtiene:

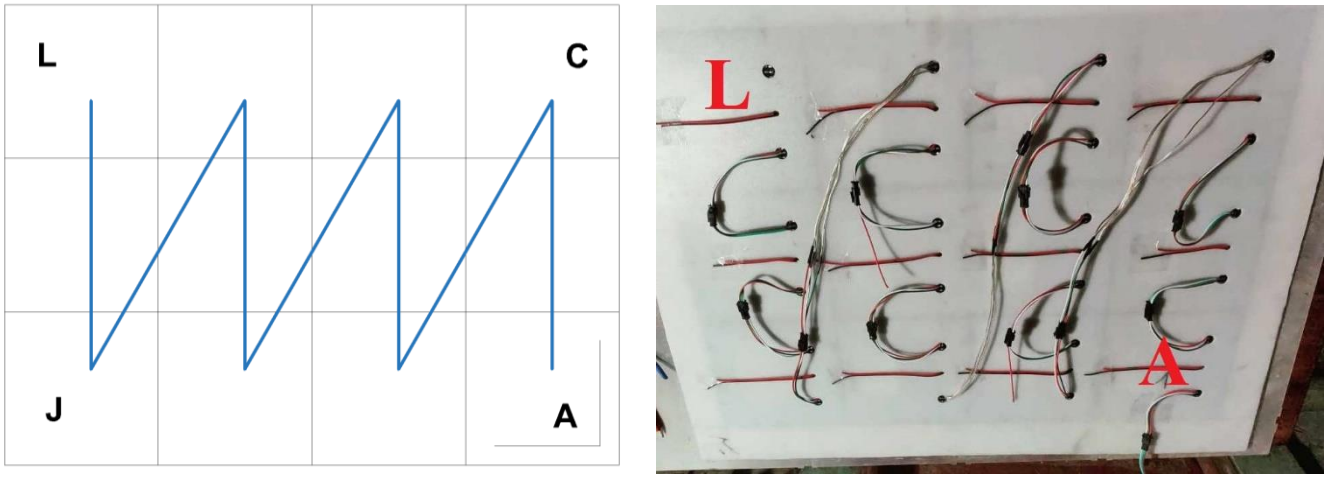
$$S = \frac{(1.5)(1m)(20.48A)}{(5.5v * 0.03)(56\ m/\Omega * mm^2)}$$

$$S = 3.3246\ mm^2$$

En este caso, de acuerdo a la **Tabla 6** se selecciona un cable de 3.5 mm^2, siendo este un cable número 12 para cada derivación que suministrará energía a todas las matrices de la pantalla.

5.3.6.2 Red de transferencia de datos.

Para la red de datos se empleó la conexión mosaica progresiva para simplificar el cableado, esto se debe configurar en el código del control de las matrices. En la **Figura 30** se muestra la red de datos ensamblada.



a) **b)**

Figura 30. a) Ejemplificación de la red en formato progresivo, b) Aplicación de la red de datos en el prototipo.

5.3.6.3 Código de controlador

Para controlar las matrices tanto de forma vertical (ESP32) como de forma horizontal (Arduino Due) se debe realizar el código en formatos diferentes, se usa el mismo lenguaje de programación para los dos dispositivos de control. Según la orientación de la pantalla la transferencia de datos debe acondicionarse.

El código para el control de las matrices debe considerar algunos parámetros que le indican a la librería como están dispuestas las matrices en la pantalla, como se trata de las mismas matrices en toda la pantalla esto se debe identificar para una sola matriz. Al momento de crear y configurar una nueva matriz, en la misma se debe declarar los siguientes parámetros:

Parámetro 1 = Ancho de cada matriz (no de la pantalla)

Parámetro 2 = Altura de cada matriz.

Parámetro 3 = Número de matrices dispuestas horizontalmente (Total de la pantalla).

Parámetro 4 = Número de matrices dispuestas verticalmente (Total de la pantalla).

Parámetro 5 = Número pin de salida de datos en el hardware (la mayoría son válidos).

Parámetro 6 = Diseño de matriz (suma de criterios). Al tratarse de LEDs en cadena, existen variables adicionales que aún no se consideran, como la posición del primer LED, la forma de la secuencia de la cadena, etc.; dichas variables deben explicarse dentro del parámetro seis, que dependiendo de la forma en la que se colocaron las matrices se deben ir sumando o quitando criterios para que la librería pueda interpretar como están dispuestos todos los LED.

A continuación, se explica cada una de estas variables y como deben digitarse en el código fuente.

- Posición del primer LED en la primera matriz, se debe elegir dos: arriba, abajo, izquierda y derecha *(NEO_MATRIX_TOP, NEO_MATRIX_BOTTOM, NEO_MATRIX_LEFT, NEO_MATRIX_RIGHT).* Un ejemplo claro para la esquina inferior derecha seria: *NEO_MATRIX_BOTTOM + NEO_MATRIX_RIGHT.*
- Indicar si los LED dentro de cada matriz están dispuestas en filas (horizontal) o en columnas (vertical), elegir una opción: *NEO_MATRIX_ROWS, NEO_MATRIX_COLUMNS.*
- Señalar si todas las filas/columnas dentro de cada matriz proceden en el mismo orden, o las líneas alternas cambian de dirección; elegir una opción: *NEO_MATRIX_PROGRESSIVE, NEO_MATRIX_ZIGZAG*. Un ejemplo de matrices que siguen el mismo orden es el de la **Figura 30a.**
- Posición de la primera matriz (mosaico) en la pantalla general; elegir dos opciones: *NEO_TILE_TOP, NEO_TILE_BOTTOM, NEO_TILE_LEFT, NEO_TILE_RIGHT.*
- Las matrices en la pantalla general están dispuestas en filas horizontales o en columnas verticales, respectivamente; elegir una opción: *NEO_TILE_ROWS, NEO_TILE_COLUMNS.*
- Las filas/columnas de matrices (mosaicos) en la pantalla general proceden en el mismo orden para cada línea, o las líneas alternas cambian de dirección; elegir uno *(NEO_TILE_PROGRESSIVE, NEO_TILE_ZIGZAG)*. Cuando se utiliza el orden en zigzag, la orientación de las matrices en filas alternas se rotará 180 grados (esto simplifica el cableado).

Aplicando todos estos parámetros, se tiene como parte del código para este proyecto (**Figura 31**), la información que se expone a continuación:

```
  FastLED NeoMatrix *matrix = new FastLED NeoMatrix (16, 16, 4, 3,
1   , PIN, NEO_MATRIX_BOTTOM     + NEO_MATRIX_LEFT +
2     NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG +
3     NEO_TILE_BOTTOM + NEO_TILE_LEFT + NEO_TILE_COLUMNS +
4 NEO_TILE_PROGRESSIVE);
```

Figura 31. Código de lectura de matrices con la librería FastLED para la pantalla.

5.3.6.3.1 Preparación de imágenes con el software Image Converter (UTFT)

Para proyectar imágenes estas se deben convertir mediante la herramienta Image Converter UTFT para obtener el código de la imagen (Ejemplo en el **Anexo**), ser cargado en el código y luego ser llamado para ser proyectado, el procedimiento para cargar una imagen en el Arduino DUE se puede observar en la **Figura 32**. Es importante destacar que la resolución máxima de las imágenes a proyectar debe ser de 48x48 pixeles.

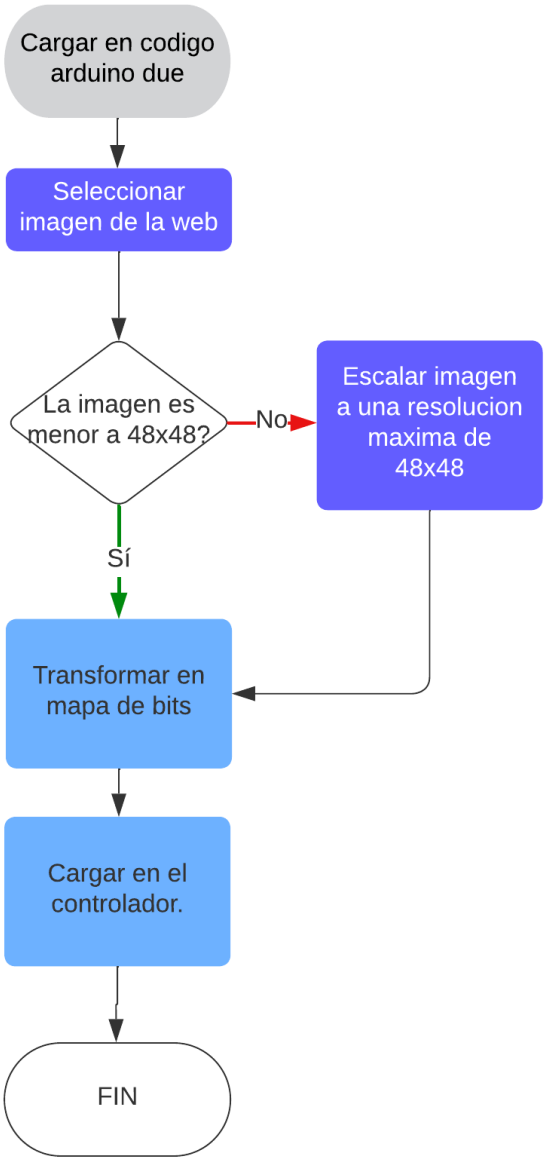


Figura 32. Pasos a seguir para cargar una imagen.

5.3.6.3.1 Arduino Due

Para este controlador se emplea la librería de FastLED para el control de matrices LED del tipo *addressable led* sin señal de reloj. Al no poseer ningún módulo adicional simplemente se debe ejecutar el código para la proyección de color, proyección de texto con algunos efectos y la proyección de imágenes.

5.3.6.3.2 ESP 32 WROOM 32S

En este controlador también se emplea la librería FastLED para el control de matrices LED del tipo *addressable led* sin señal de reloj. Con este controlador se usarán algunos módulos adicionales como una pantalla para la selección de los registros que se graban en una microSD (pantalla y lector de tarjetas microSD).

5.3.7 Validación del prototipo construido

5.3.7.1 Preparación del prototipo.

Antes de iniciar con cualquier proyección se debe energizar el prototipo para estabilizar la fuente de poder, girar la pantalla de acuerdo al controlador a usarse, para después seleccionar el controlador a utilizar.

En el caso del seleccionar el controlador de Arduino Due este reproducirá el programa automáticamente de forma secuencial; primero el texto que se cargó en el código y luego las imágenes.

Cuando se utiliza el controlador esp32 se desplegará un menú donde están las animaciones y configuraciones como el brillo y velocidad. Para desplazarse en el menú se debe usar el pulsador rotatorio EC11, al seleccionar una animación se reproducirá en un bucle infinito hasta que se cambie de animación. El proceso de funcionamiento de ambos controladores, se puede observar en el diagrama de flujo de funcionamiento expuesto en la **Figura 33**.

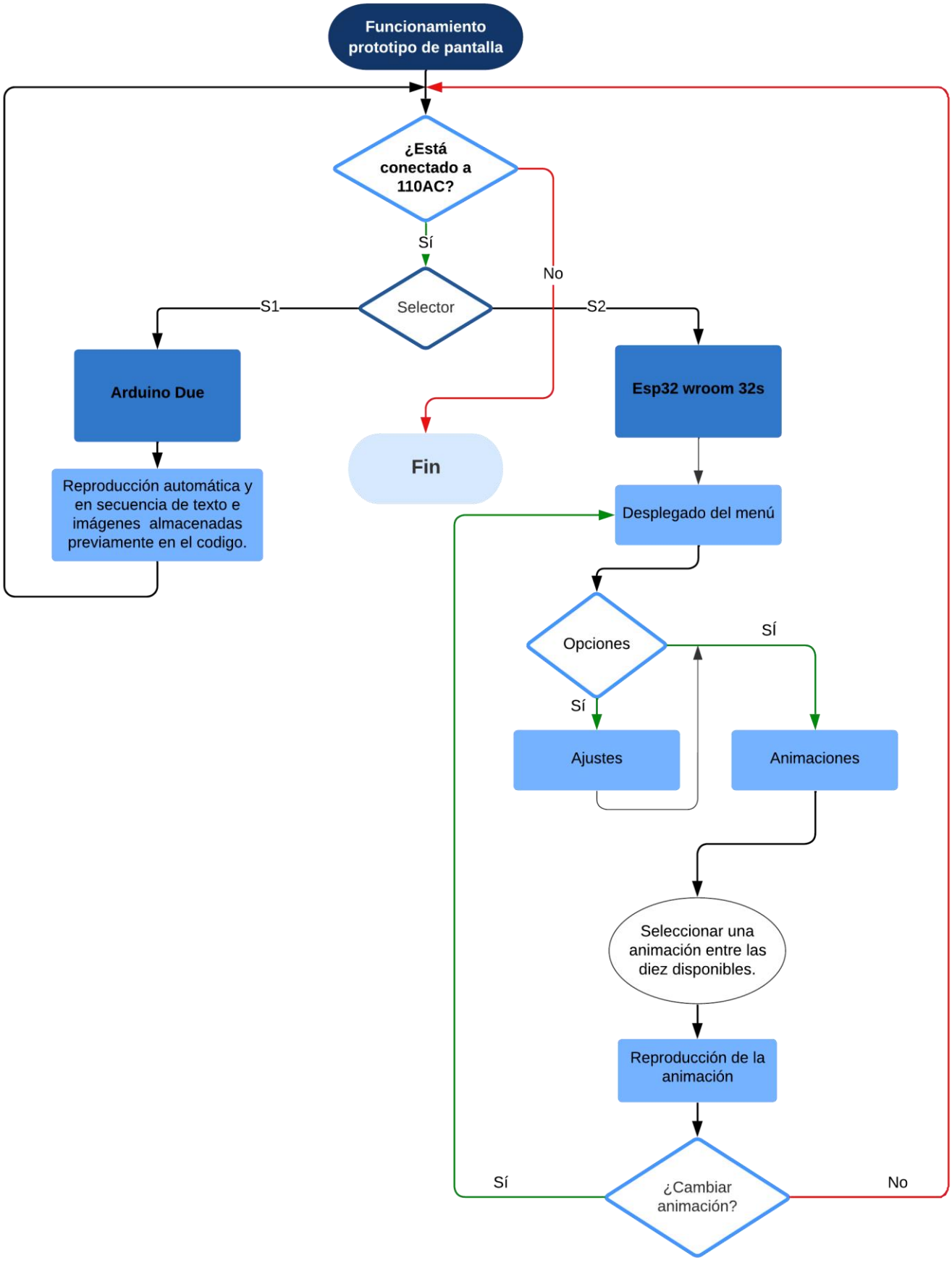


Figura 33. Diagrama de flujo del funcionamiento del prototipo de pantalla.

5.3.7.2 Proyección de texto.

La proyección de texto con el controlador de Arduino DUE es de forma automática, para cambiar el texto a proyectar este se debe configurar en el código fuente y volver a cargar en el controlador. En el controlador esp32 se selecciona la animación correspondiente en el menú desplegable. En la **Figura 34** se muestra la proyección de texto para cada controlador. El código para cada caso se expone en el **Anexo** y **Anexo** respectivamente.

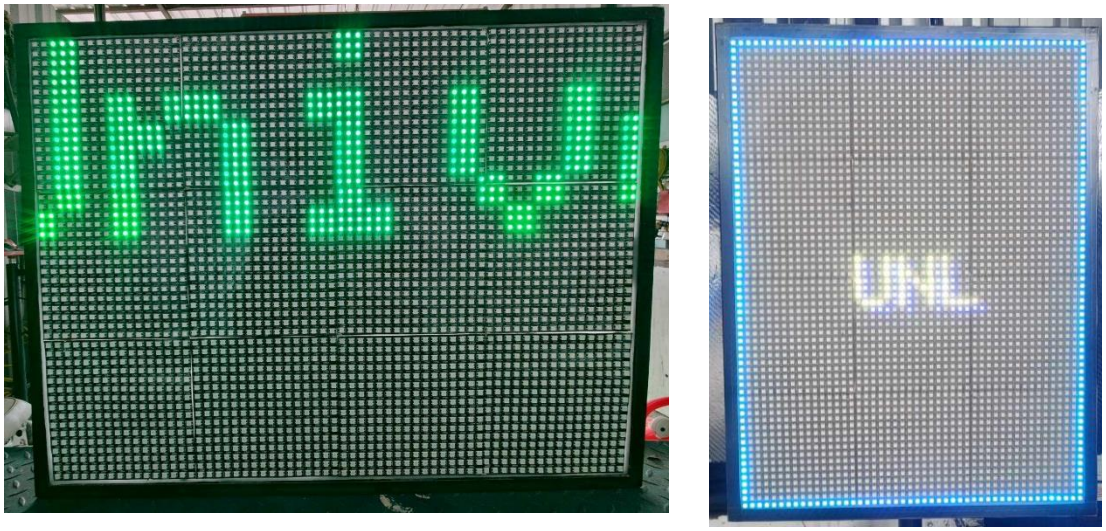


Figura 34. Proyección de texto formato scroll en Arduino y texto fijo con esp32.

5.3.7.3 Proyección de imágenes.

En la proyección de imágenes se debe considerar el tamaño de la imagen y el formato que admite el código. A continuación, en la **Figura 35** se muestran dos ejemplos de imágenes, en el **Anexo** se expone el código completo para cada caso.

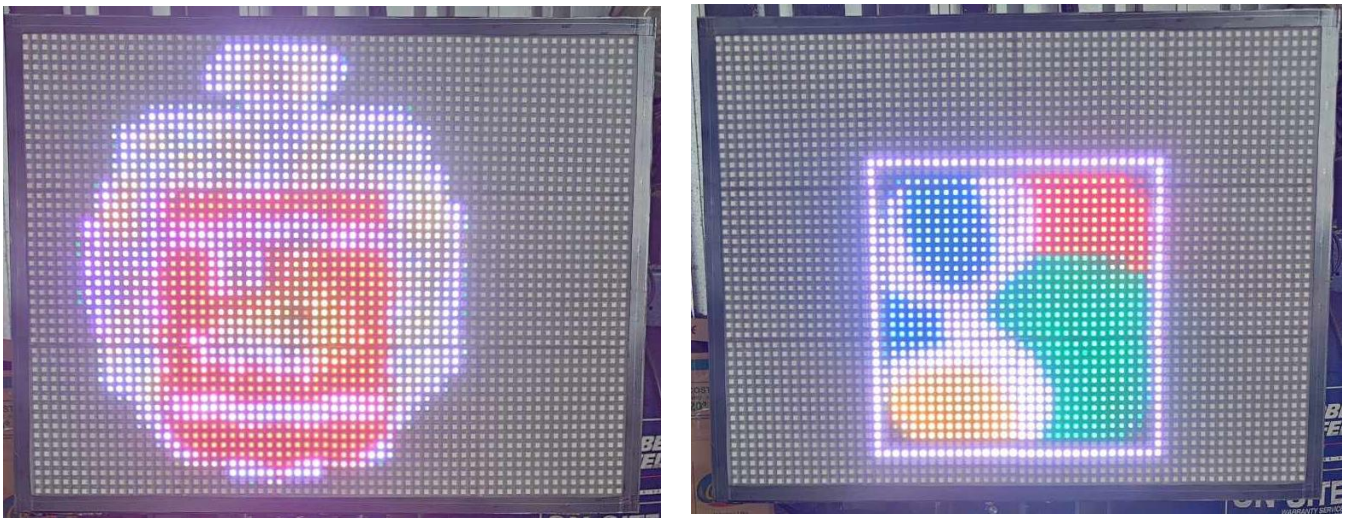


Figura 35. Representación del escudo de la provincia de Loja de 48x48 pixeles y parte del logo Google de 32x32 pixeles.

6. Resultados

6.1 Análisis de los tipos de matrices LED.

La construcción del prototipo de pantalla informativa basada en matrices LED RGB se desarrolla mediante un proceso secuencial que considera la selección de componentes y ensamblaje de materiales y equipos. Luego de la revisión de literatura técnica para la selección de cada uno de los materiales y componentes que conforman el prototipo de pantalla, se selecciona las características de los siguientes equipos y materiales que se exponen en la **Tabla 13**, se presenta la valoración total obtenida para cada tipo de matriz LED considerando los criterios expuestos y según la matriz de ponderación expuesta en la **Tabla 12**, donde se observa que la matriz con mayor puntación es la WS2812B.

Tabla 13. Valoración de cada matriz LED a partir de la matriz de ponderación.

Técnica	***WS2801***	***Matriz-punto***	***Panel smd***	***WS2812B***
Valoración total	81	64,6	78	**91,8**

La matriz LED RGB WS2812B fue seleccionada debido a su alta calidad visual y funcionalidad del dispositivo (como la calidad del color, el brillo y la resolución) y la facilidad de control, además es un dispositivo versátil utilizado en variedad de aplicaciones que incluyen pantallas informativas a proyectos de iluminación y arte interactivo. El tamaño de las matrices seleccionadas es de 16x16 cm, las dimensiones del prototipo propuesto se pueden observar en la **Figura 24.**

6.2 Diseño y construcción de prototipo

6.2.1 Diseño del prototipo

Para el diseño del prototipo, es necesario iniciar con la selección de un dispositivo de control para las matrices LED, para continuar con el dimensionamiento de la red de distribución de energía y de datos. A continuación, se expone el análisis detallado de cada etapa de selección y construcción del prototipo.

- **Controlador de las matrices LED**

En la **Tabla 14** se muestran los valores obtenidos para los tres controladores presentados en la **Tabla 11** y **Tabla 12**, los cuales se evaluaron según sus ventajas y desventajas, y criterios de ponderación basados principalmente en la potencia de procesamiento, facilidad de uso, conocimiento previo y costo.

Tabla 14. Valoración total para cada controlador según la matriz de ponderación elaborada.

Controlador		Valoración total
1		59
2		**81**
3		**81**

El **controlador 2 y 3** presentan la valoración más alta y de igual valoración, teniendo en cuenta que ambos controladores interpretan el mismo lenguaje de programación y software IDE de Arduino; además de los criterios de ponderación expuestos en la ¡Error! No se encuentra el origen de la referencia.. Por lo tanto, la aplicación de ambos controladores en dos diferentes formatos (horizontal y vertical) permitirá además contrastar su uso. En el prototipo para seleccionar cualquiera de los dos controladores se acondiciona en un dispositivo que permite la selección de funcionamiento de cada uno de estas dos variantes de control. Existen bibliotecas de software disponibles, como FastLED y Adafruit NeoPixel, que simplifican el proceso de programación y control de estas matrices. Mediante el envío de datos seriales desde el microcontrolador a la matriz, es posible actualizar el color de cada LED, lo que permite la creación de efectos y animaciones visuales.

- **Diseño y dimensionamiento de la estructura del prototipo**

Para el dimensionamiento de la estructura del prototipo se consideró, el tamaño de cada matriz y el total de las mismas, se diseña los marcos de soporte de la pantalla para formato vertical y horizontal, se consideró un sistema de eje para poder girar la pantalla.

- **Diseño y dimensionamiento de la red de distribución de energía y datos**

La red de distribución de energía del prototipo de pantalla inicia desde la fuente de poder hasta la entrada a cada matriz individual para abastecer a los LEDs de energía, su diseño y dimensionamiento depende de las necesidades de la pantalla en general, con base en la demanda de cada proyección dada, ya sea de texto o imagen. En la **sección 5.3.4** se determina el requerimiento de energía mínimo de la pantalla, siendo este de 61,44 A. Considerando que el consumo de todas las matrices es muy elevado, se utilizan tres derivaciones (paralelo) de

conexión para alimentar a todas las matrices con 5V DC, cada derivación energiza a 4 matrices, como se puede observar en la **Figura 28.**

En el caso de la transferencia de datos, las condiciones de diseño son diferentes, ya que el control se realiza de forma secuencial, tipo "cadena", no se utilizan derivaciones (serie); por lo tanto, se consideró un formato secuencial para la conexión y transferencia de datos. Es así que se considera para el prototipo un circuito de fuerza que alimenta los LEDs y otro de control que suministra el controlador (Arduino DUE & ESP32) para la configuración de eventos, en la **Figura 36** se exponen la configuración del circuito de fuerza y control.

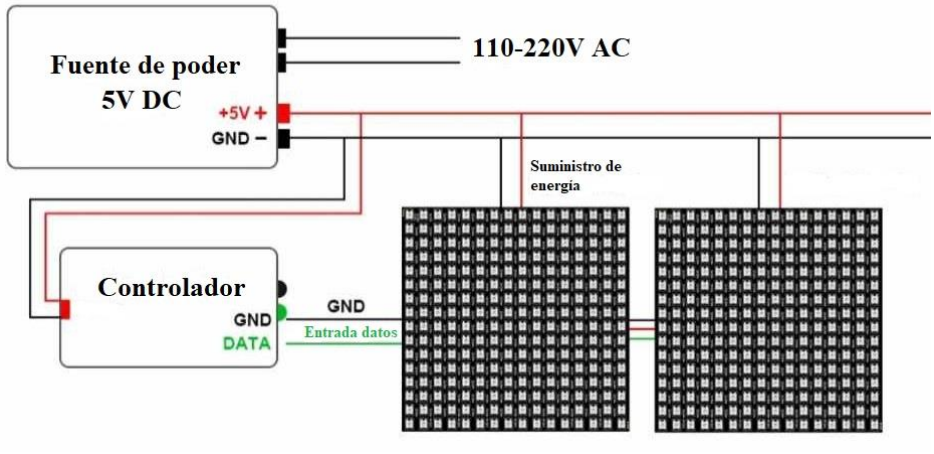


Figura 36. Diseño del circuito de fuerza y control.

Para la transferencia de datos y considerando el código del programa, la librería permite múltiples formas para conectar las matrices. En este prototipo se implementó la conexión secuencial (ver **Figura 30**), en el **Anexo 5** se puede observar algunas otras posibles configuraciones.

- **Dimensionamiento de cables.**

El cálculo de la sección del cable a utilizar en las matrices LED RGB WS2812B depende principalmente de la corriente de consumo del prototipo, considerando un valor de 61,44A para toda la pantalla**,** además se debe considerar la longitud de la conexión entre las matrices y la fuente de alimentación.

Por lo tanto, la longitud del cable para cada derivación es de 1 m con una corriente de 20.48 A por las tres derivaciones a 5 V. Aplicando la ecuación 1 para la sección del cable, se obtiene:

$$S = \frac{(1.5)(1m)(20.48A)}{(5.5v * 0.03)(56\ m/\Omega * mm^2)}$$

$$S = 3.3246\ mm^2$$

En este caso, de acuerdo a la **Tabla 6** se selecciona un cable de 3.5 mm^2, siendo este un cable número 12 para cada derivación que dará energía a todas las matrices de la pantalla.

- **Selección de la fuente de poder**

La fuente de poder JPS300V que se selecciona para el prototipo construido es de **60A** (ver **Anexo D**), cumpliendo así los requerimientos de energía.

- **Programación**

Para la arquitectura de control se empleó un sistema de matriz individual para controlar cada uno de los LEDs que conforman la pantalla, cada LED forma un píxel. Este sistema modular consiste en varios niveles, cada nivel tiene una lógica para controlar la distribución de los datos y la generación de señales de control. La señal origen proporciona, en este caso, el texto a ser mostrado, como se expone en la **Figura 37**.

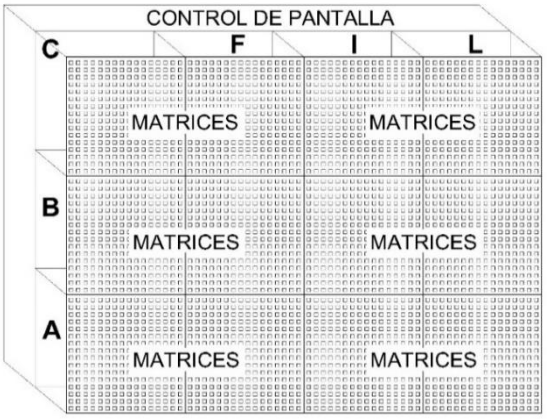


Figura 37. Arquitectura de control para el prototipo de pantalla.

En la plataforma de programación para el controlador se debe configurar la forma de conexión de las matrices para la transferencia de datos, las bibliotecas utilizan tanto matrices individuales (todos los NeoPixel en una única cuadrícula uniforme) así como matrices en mosaico (múltiples cuadrículas combinadas en una pantalla más grande).

Arduino DUE

Como se expuso, para este controlar se utiliza una configuración sin una interfaz de usuario; es en el código fuente que se deben realizar cambios y volver a cargarse. Además, con el controlador Arduino DUE la pantalla trabajará de forma horizontal, pero se puede presentar texto inclinado en formato vertical.

Para cambiar el texto que se proyecta se puede realizar en las líneas 240, 245 y 259 del código fuente que se encuentra en el **Anexo 6**. En líneas adyacentes a estas se puede cambiar el color y orientación de proyección. Para esta controladora se utiliza la librería FastLED, el código completo se encuentra disponible en el **Anexo 6**. Para la proyección de imágenes se debe escalar la misma a una resolución máxima de 48x48 pixeles y luego se debe convertir con el *convertidor de imágenes* UTFT (online en este caso) y ser llamado en el código en la línea 5 para imágenes de 48x48 pixeles (ver **Figura 38**) y en la línea 23 para imágenes de 32x32 pixeles.

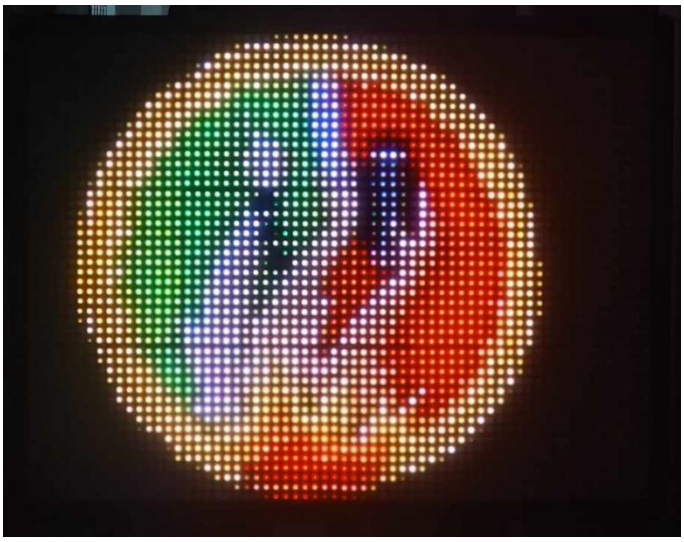


Figura 38. Proyección del logo de la Universidad Nacional de Loja.

ESP32

Al tratarse de un microcontrolador más robusto, en este caso se plantea un sistema o interfaz para interactuar con el usuario. En la programación de eventos se considera el uso de una tarjeta SD donde se cargarán las imágenes a proyectar, y mediante un codificador y una pantalla LCD se puede visualizar y seleccionar el programa, la secuencia de funcionamiento para esta variante se expone en el diagrama de flujo de la **Figura 39**. Además, está variante de funcionamiento puede funcionar en tiempo real haciendo uso de una interfaz de uso específico que proporciona este equipo, se puede proyectar los eventos configurados directamente en el prototipo de pantalla.

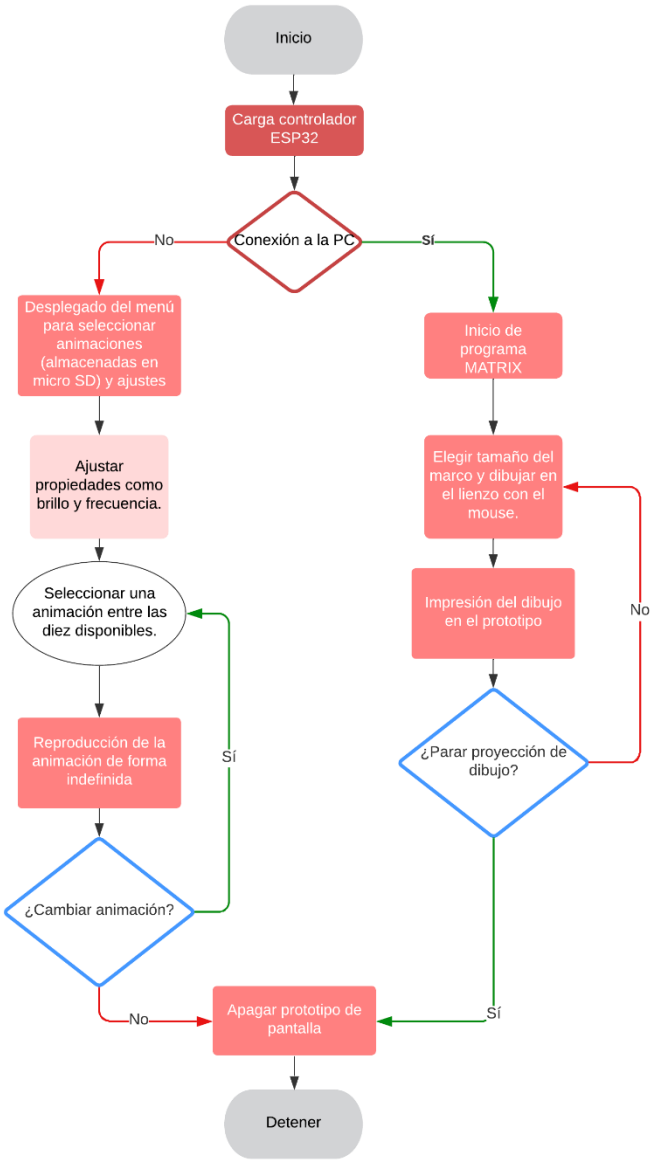


Figura 39. Diagrama de flujo del sistema con el controlador ESP32.

Para este modo de funcionamiento la pantalla se debe girar 90° en el sentido de las agujas del reloj, para que esta se ubique de forma vertical, como se puede observar en la **Figura 40**. El código de este controlador se encuentra en el **Anexo 7**.

Figura 40. Pantalla vertical para ESP32.

- **Prototipo construido**

En la **Figura 41** se puede observar la implementación del prototipo de pantalla informativa con matrices LED RGB propuesto en los objetivos del proyecto.

Figura 41. Prototipo de pantalla informativa construido. Presentación del formato horizontal, vertical y parte posterior de la pantalla.

- Costos referenciales

  En esta sección se describen el costo referencial de los materiales utilizados para la construcción del prototipo.

Descripción/componente	Cantidad	Precio UNIT. $	V/TOTAL $	Proveedor
Diodo rectificador 800V 1N4006	1	0,05	0,05	Electrónica Digital Jaramillo
Display OLED I2C 0.96inch white 64x128	1	7,5	7,5	Mega Electronics
ESP32 Dev Kit ESP-WROOM-32S	1	8	8	Aliexpress
Matriz led flexible RGB WS2812B 16x16	13	8	104	Aliexpress
Pulsador de interruptor de código de codificador rotatorio de 360 grados EC11	1	3	3	Aliexpress
Cable dupont H/M H/H M/M	1	5	5	Aliexpress
Micro SD Card Module Memory Shield	1	1,4	1,4	Aliexpress
8 GB memory card	1	10	10	Nintenlandia
Fuente de alimentación JPS300V- 5V 180A 900W	1	60	60	Aliexpress
Arduino DUE	1	20	20	Aliexpress
Cable #12 de cobre	6	0,45	2,7	Ferreteria Nacio.
Terminales tipo herradura	10	0,1	1	Ferreteria Nacio.
Adaptador ESP32	1	3	3	Aliexpress
Marco de madera	1	10	10	Carpinteria Nacio.
1/32 Pintura negra esmalte	1	2	2	Ferreteria Nacio.
Caja Tornillos Drywall Screws	1	1	1	Ferreteria Nacio.
Transporte	1	50	50	Macara Express
Mano de obra	1	50	50	Autor
		Total	**$338,65**	

7. Discusión

La investigación de este proyecto se ha centrado en construir y validar un prototipo de pantalla informativa utilizando matrices LED RGB que se pueda implementar en áreas urbanas para transmitir diferentes tipos de mensajes informativos de manera dinámica.

Eficiencia y Flexibilidad del Sistema:

Existen varios tipos de tecnología LED aplicables a pantallas informativas; sin embargo, de acuerdo a los resultados del análisis de las características de cada tipo de tecnología LED, se obtuvo que la más adecuada para los propósitos del proyecto es el LED WS2812B con un 10% de ventajas frente a otros tipos de LEDs RGB. Utilizar estas matrices permite la visualización de contenido y una eficiencia energética aceptable, la capacidad de controlar cada LED individualmente ha permitido la creación de efectos visuales dinámicos. De acuerdo a Vazquez (2019) el uso de los LED WS2812B además permite evitar técnicas de barrido ya que esta tecnología tiene la capacidad de retener los datos y permite alcanzar mayores niveles de brillo y de color, sugiriendo ser una opción muy viable para este tipo de pantallas informativas. Esto demuestra que el LED WS2812B es una tecnología óptima que puede aplicarse en pantallas para difundir información en entornos urbanos.

En cuanto al diseño del prototipo, la elaboración de una matriz de ponderación y el análisis de las ventajas y desventaja de los tres controladores, facilita la selección del hardware adecuado para el control de matrices LED; obteniendo así, para el controlador Arduino Due un 81 % de correlación con las características requeridas; sin embargo, el tercer controlador presenta mayores recursos que aumentan la potencia de procesamiento que es importante para la proyección de imágenes de mayor resolución. Es así que el controlador ESP32 también tiene 81% de correlación para las características requeridas y planteadas. De acuerdo a Soto, Soto, & Vásquez (2012) para el control de una pantalla de LEDs se puede usar un microcontrolador para enviar datos a un decodificador de columnas, pero resulta mucho más rápido el uso de un hardware programable con sistemas de señal PWM. De esta manera se determina que el controlador dos y tres (Arduino Due y Esp32), son los más conveniente por los requerimientos del prototipo de pantalla requerido.

Con respecto al diseño de la red de distribución de energía y datos, al tener una estructura que debía considerar un formato vertical y horizontal hacían que el cableado deba ser dinámico, añadiendo más complejidad. Y esto fue aún más evidente al considerar tres derivaciones para energizar todas las matrices, ya que esto representaba más cables en movimiento. Para el dimensionamiento de los cables, es necesario considerar el consumo

teórico de la toda la pantalla que es de 184,32A; pero de acuerdo a Adafruit (2023) en su página web menciona que en este tipo de LED su consumo energético no será del 100%, ya que depende totalmente de lo que se esté proyectando. Se sugiere considerar un consumo de $1/3$ del total como regla general sin tener efectos nocivos y sin perder la calidad de brillo en ciertos escenarios específicos. Por lo tanto, el consumo total de la pantalla es de 61,44A. Aun así, este valor es elevado, se plantea considerar derivaciones de distribución para segmentar el consumo de energía, se seleccionó una fuente de poder de 60A por salida, debido a que comercialmente es la más aproximada a los parámetros requeridos por el sistema. Harwood (2017) en su investigación presenta el uso de baterías para el control de LEDs, pero establece que deben ser de grandes dimensiones, siendo necesario el uso de un convertidor DC/DC. Además, en el tema del cableado y conectores establece una solución similar de derivaciones y establece que es térmica y eléctricamente seguro.

Aunque Arduino y ESP32 son una opción viable como controlador para el prototipo de pantalla informativa basada en matrices LED RGB WS2812B. En lo que respecta a la facilidad de programación, el Arduino Due tiene una ventaja en términos de familiaridad y compatibilidad con el entorno de desarrollo de Arduino, que es ampliamente utilizado y cuenta con una gran cantidad de recursos y bibliotecas disponibles. Esto facilita la programación y el desarrollo de proyectos para usuarios familiarizados con el ecosistema de Arduino. Sin embargo, el controlador ESP32 también cuenta con un entorno de desarrollo robusto y una amplia gama de bibliotecas disponibles, lo que lo hace igualmente accesible para los desarrolladores.

Optimización del Hardware y Software:

Los controladores seleccionados en cuanto a hardware y software han contribuido a maximizar el rendimiento y la estabilidad del prototipo. Sin embargo, se reconoce que aún existen oportunidades para mejorar la eficiencia y la escalabilidad del sistema mediante la implementación de técnicas de control adicionales.

Experiencia del Usuario y Aplicaciones Potenciales:

La pantalla informativa desarrollada permite a los usuarios proyectar una variedad de entornos. La secuencia de programación de eventos es amigable con el usuario para la proyección de información. Sin embargo, es importante considerar las necesidades y preferencias del usuario final al diseñar y desplegar sistemas de visualización.

8. Conclusiones

Se realizó un exhaustivo análisis de las tecnologías LED RGB disponibles en el mercado, identificando sus ventajas, desventajas y aplicaciones específicas. Este estudio permitió seleccionar las matrices LED RGB más adecuadas para la construcción de la pantalla informativa, teniendo en cuenta factores como calidad de la proyección de colores, facilidad de control y la durabilidad aproximada de 50000 horas de trabajo.

Para el proceso de selección se llevaron a cabo los cálculos para dimensionar los componentes y materiales necesarios con el tipo de LED RGB WS2812B, cuyas dimensiones para el prototipo fueron de 64 x 48 cm, además con esto se incluyó la selección de controladores (Arduino Due y ESP 32), fuente de alimentación, y estructuras de soporte, asegurando que el prototipo sea escalable y adaptable a diferentes necesidades y tamaños, usando 12 matrices LED. La elección adecuada de estos elementos fue crucial para garantizar la funcionalidad y la estabilidad del sistema. Destacando que la fuente de poder presento ser un reto en cuanto los requerimientos del sistema, teniendo como resultado final una fuente de poder JP300V de 5VDC y 300W por salida. Al ser escalable a diferentes dimensiones, se debe recalcular la fuente de alimentación y demás componentes para garantizar el funcionamiento, al escalar las dimensiones del prototipo se debe ajustar los parámetros de programación configurados inicialmente.

Se construyó y validó un prototipo funcional de pantalla informativa utilizando matrices LED RGB. El desarrollo de código ha sido llevado a cabo de manera estructurada y eficiente, garantizando un funcionamiento adecuado del prototipo en todas las condiciones de uso previstas, validando su funcionamiento. Las pruebas realizadas demostraron que el prototipo es capaz de proyectar texto alfanumérico e imágenes con una calidad aceptable y una correcta visibilidad. Además, se evaluaron aspectos como la facilidad de uso, la capacidad de actualización de contenidos y la respuesta del sistema a diferentes condiciones de iluminación ambiental. Los resultados de estas pruebas confirmaron que el prototipo cumple con los requisitos iniciales y tiene el potencial para ser utilizado en diversas aplicaciones informativas.

9. Recomendaciones

Aunque el prototipo de pantalla informativa ha alcanzado los objetivos establecidos en esta investigación, es importante reconocer que aún existen algunas limitaciones y desafíos pendientes como mejorar la eficiencia energética, así como implementar una base protectora para ambientes exteriores, y la exploración de nuevas técnicas de interacción y control. Se identifican áreas específicas para futuras investigaciones, como el desarrollo de algoritmos avanzados de procesamiento de imagen y la integración de tecnologías emergentes, como la realidad aumentada y la inteligencia artificial.

Implementación de lógica condicional para activar eventos basados en criterios como la temperatura ambiente, el nivel de luz o la presencia de movimiento.

10. Bibliografía

adafruit. (28 de Noviembre de 2023). *The Magic of NeoPixels*. Obtenido de https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels

Alibaba. (22 de Noviembre de 2023). *RGBCCT*. Obtenido de https://ae01.alicdn.com/kf/H18c5bde1eaa64aa4b8fa3f14d0b9353aR.jpg

Arduino. (19 de Noviembre de 2023). Obtenido de LED RGB: http://ceca.uaeh.edu.mx/informatica/oas_final/OA4/led_rgb.html

ARDUINO. (30 de Noviembre de 2023). *Arduino DUE*. Obtenido de https://store-usa.arduino.cc/products/arduino-due?selectedStore=us

ART ROCKET. (19 de Noviembre de 2023). Obtenido de Qué son RGB y CMYK y cuándo usar cada uno: https://www.clipstudio.net/aprende-a-dibujar/archives/157955

COELECTRIX Corporation. (15 de Enero de 2024). *Dimensionamiento y selección de los cables*. Obtenido de https://coelectrix.com/blog/dimensionamiento-y-seleccion-de-los-cables

DINALIGHT. (21 de Noviembre de 2023). *Pantallas LED* . Obtenido de https://www.dinalight.com/pantallasled/?campaignid=17789209896&adgroupid=134244927730&keyword=pantallas%20para%20publicidad%20exterior&device=c&loc_physical_ms=1005375&loc_interest_ms=&gad_source=1&gclid=CjwKCAiA98WrBhAYEiwA2WvhOgvCfbaoxqKvZpoAd4Bq0-zGRyj

ESPRESSIF. (30 de Noviembre de 2023). *ESP32*. Obtenido de https://www.espressif.com/en/products/socs/esp32

Figueroa, T. (2017). *Diseño de un letrero LED programable utilizando microcontroladores PICs en la facultad de ciencias técnicas de la Universidad Estatal del Sur de Manabi.* Jipijapa: UNESUM.

Gago, A. (2012). *Iluminación con tecnología LED.* Paraninfo.

García, P. (2016). *Estudio técnico, económico y medioambiental de una instalación renovable aislada de la red eléctrica.* Zaragoza: Escuela de Ingeniería y Arquitectura. Universidad Zaragoza.

Harwood, S. (2017). *Diseño y construcción de un controlador LED con múltiples modos de funcionamiento.* Madrid: UPM.

Houkem. (19 de Noviembre de 2023). *Pantalla de matriz de puntos LED RGB de 64x64mm 16x16*. Obtenido de http://es.dghongke.com/product/604x64mm-16x16-rgb-led-dot-matrix-display

Huang, Y., Tan, G., Gou, F., Li, M.-C., Lee, S.-L., & Wu, S.-T. (2019). *Prospects and challenges of mini-LED and micro-LED.* Orlando: WILEY.

LED. (30 de Noviembre de 2023). *Procesador de video LED para pantalla LED.* . Obtenido de https://led-control.com/es/the-definition-of-led-video-processor-for-led-display-screen/

Lenovo. (23 de Noviembre de 2023). *¿Cómo mejoran las pantallas OLED nuestros dispositivos?* Obtenido de https://www.bloglenovo.es/como-mejoran-las-pantallas-oled-nuestros-dispositivos/

Llamas, L. (30 de Noviembre de 2023). *Conectar Arduino con paneles y tiras LED RGB WS2812B (NeoPixel)*. Obtenido de https://www.luisllamas.es/arduino-led-rgb-ws2812b/

López, E. (2017). *Raspberry Pi. Fundamentos y aplicaciones.* Madrid: RA-MA.

Medrano, E. (2010). *Rediseño e implementación de un sistema de iluminación para espacios publicitarios usando LED RGB.* Lima : PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ.

MegaLámparas. (18 de Noviembre de 2023). Obtenido de ¿Qué son los diodos LED? Conoce sus características: https://megalamparas.com.gt/diodos-led-que-son/#:~:text=Los%20diodos%20LED%20proporcionan%20ventajas,LED%20es%20la%20mejor%20decisi%C3%B3n.

Mocholí, A., Borja, A., & Sánchez, M. (21 de Noviembre de 2023). *Tipos de pantallas publicitarias: LED, LCD, plasma, OLED y proyectores*. Obtenido de https://www.emprendedorenlanube.com/analisis-productos/pantallas-publicitarias.php

Orient Display. (23 de Noviembre de 2023). *¿Cómo funciona LCD?* Obtenido de https://www.orientdisplay.com/es/knowledge-base/lcd-basics/how-liquid-crystal-displays-work/

Phoonkotchakorn, T. (2019). *RASPBERRY PI.* Bangkok: V.Print.

Pimputkar, S., Speck, J., DenBaars, S., & Nakamura, S. (2009). *Prospects for LED lighting.* California: Macmillan.

PipBricks. (16 de Noviembre de 2023). *Etsy*. Obtenido de https://www.etsy.com/es/listing/1068635129/pixel-mosaic-art-mona-lisa-de-da-vinci

Solas, P. (2014). *Teoría general de la imagen.*

Soto, J., Soto, D., & Vásquez, F. (s.f.). *Sistema de información visual* . Quito: Escuela Politécnica Nacional .

Suárez, C. (2013). *Implementación de un sistema de información para la FIEE utilizando módulos de LEDs RGB.* Quito: Escuela Politécnica Nacional .

TALLERELECTRONICA.COM/BLOG. (18 de Noviembre de 2023). Obtenido de DIODO LED: https://tallerelectronica.com/diodo-led/

Vazquez, M. (2019). *Diseño y control electrónico de una matriz de LEDs RGB para la proyección de imágenes y texto alfanumérico.* Cuernavaca: UAEM.

Worldsemi. (30 de Noviembre de 2023). *WS2812B Intelligent control LED integrated light source*. Obtenido de https://html.alldatasheet.com/html-pdf/1179113/WORLDSEMI/WS2812B/2858/5/WS2812B.html

Yam, F., & Hassan, Z. (2004). *Innovative advances in LED technology*. Malaysia: ELSEVIER.

Zendi, F. (2015). *PERANCANGAN SCORE BOARD DAN TIMER MENGGUNAKAN LED* . Jakarta Barat: SINERGI.

11. Anexos

Anexo 1. Tipos de pantallas LED

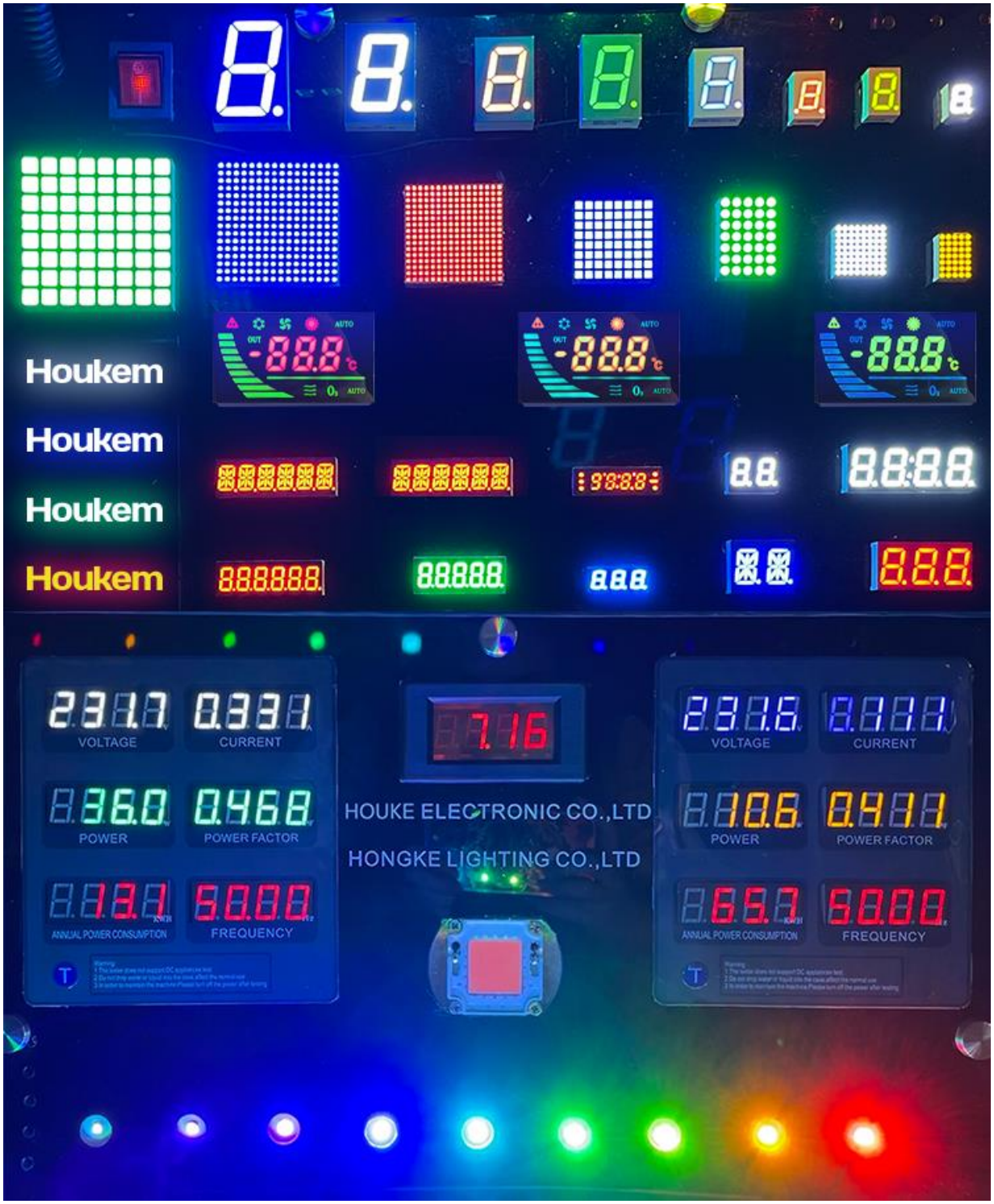


Figura 42. Tipos de indicadores LED.

Fuente: (Houkem, 2023)

Anexo 2. Características Arduino Due

Tabla 15. Características del ARDUINO DUE.

Board	
Name	Arduino® Due
SKU	A000062
Microcontroller	
AT91SAM3X8E	
USB connector	
Micro USB	
Pins	
Built-in LED Pin	13
Digital I/O Pins	54
Analog input pins	12
Analog output pins	2
PWM pins	12
Communication	
CAN	Yes (ext. transceiver needed)
UART	Yes, 4
I2C	Yes
SPI	Yes
Power	
I/O Voltage	3.3V
Input voltage (nominal)	7-12V
DC Current per I/O pin (group 1)	9 mA
DC Current per I/O pin (group 2)	3 mA
Power Supply Connector	Barrel Plug
Total DC Output Current on all I/O lines	130 mA
Clock speed	
Processor	AT91SAM3X8E 84 MHz

Fuente: *(ARDUINO, 2023)*

Anexo 3. Características Esp32

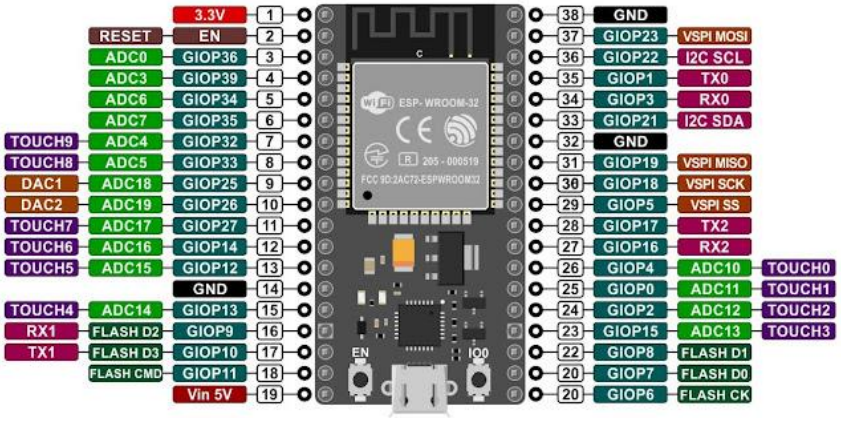


Figura 43. Pinout ESP 32.
Fuente: (Asanza, 2024).

Tabla 16. Características ESP Wroom 32

240 MHz dual core Tensilica LX6 microcontroller with 600 DMIPS
Integrated 520 KB SRAM
Integrated 802.11b/g/n HT40 Wi-Fi transceiver, baseband, stack and LWIP
Integrated dual mode Bluetooth (classic and BLE)
4 MByte flash include in the WROOM32 module
On-board PCB antenna
Ultra-low noise analog amplifier
Hall sensor
10x capacitive touch interface
32 kHz crystal oscillator
3 x UARTs (only two are configured by default in the Feather Arduino IDE support, one UART is used for bootloading/debug)
3 x SPI (only one is configured by default in the Feather Arduino IDE support)
2 x I2C (only one is configured by default in the Feather Arduino IDE support)
12 x ADC input channels
2 x I2S Audio
2 x DAC
PWM/timer input/output available on every GPIO pin
OpenOCD debug interface with 32 kB TRAX buffer
SDIO master/slave 50 MHz
SD-card interface support

Fuente: (Asanza, 2024).

Anexo 4. Fuente de poder JPS300V

Figura 44. Fuente de poder JPS300V.

Anexo 5. Configuraciones en la transferencia de datos

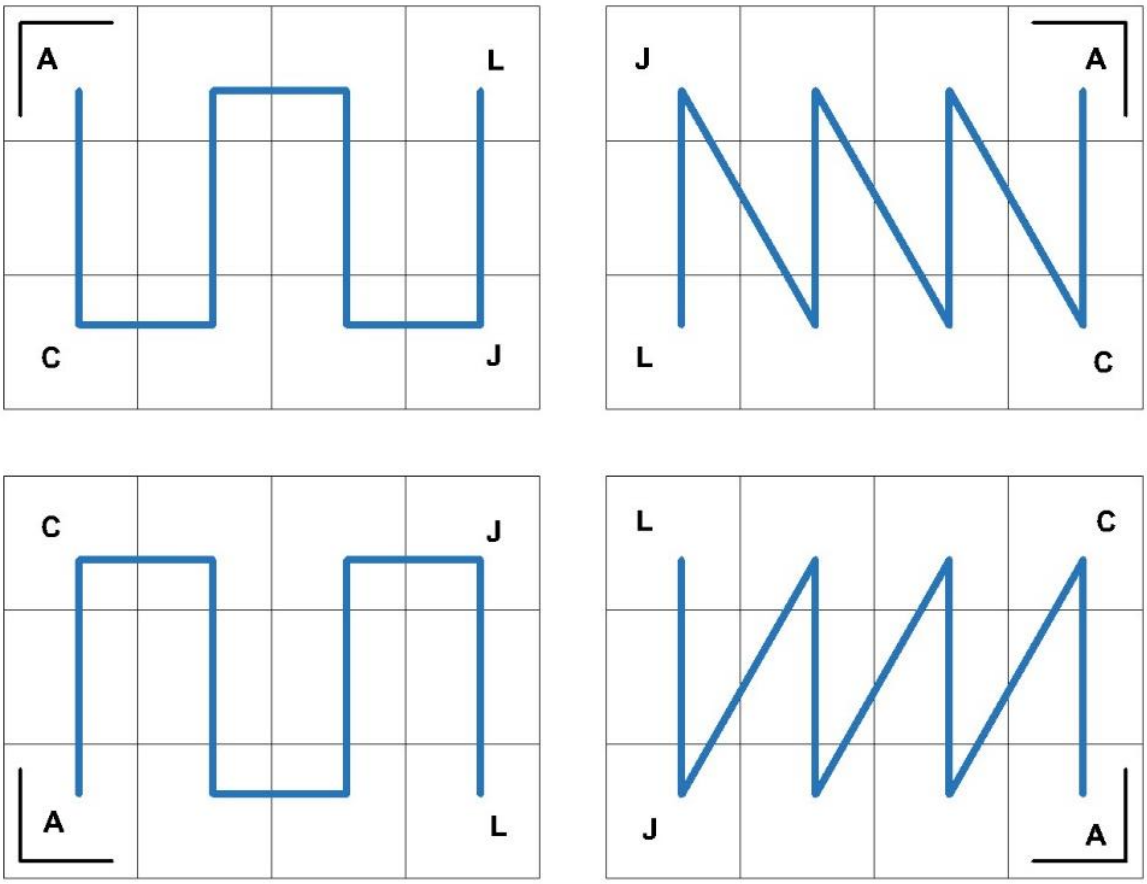


Figura 45: Tipos de algunas configuraciones para la transferencia de datos en las matrices.

Anexo 6. Código del Arduino Due

```
#include <Adafruit_GFX.h>
#include <FastLED_NeoMatrix.h>
#include <FastLED.h>

#include "logoLoja.h"

#ifndef PSTR
 #define PSTR
#endif

#define PIN 6

#ifdef ESP8266
#define PIN RX
#endif

#define P16BY16X4
#if defined(P16BY16X4) || defined(P32BY8X3)
#define BM32
#endif

#ifdef BM32
#include "google32.h"
#endif

#define BRIGHTNESS 80 // Configuracion del brillo de los LED (0-255)

#if defined(P16BY16X4)
#define mw 64
#define mh 48
#define NUMMATRIX (mw*mh)
CRGB leds[NUMMATRIX];
FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, 16, 16, 4, 3,
  NEO_MATRIX_BOTTOM     + NEO_MATRIX_LEFT +
    NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG +
```

```
    NEO_TILE_BOTTOM + NEO_TILE_LEFT + NEO_TILE_COLUMNS + NEO_TILE_PROGRESSIVE);

#elif defined(P32BY8X3)
#define mw 0
#define mh 0
#define NUMMATRIX (mw*mh)
CRGB leds[NUMMATRIX];
FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, 8, 32, 3, 1,
  NEO_MATRIX_BOTTOM     + NEO_MATRIX_LEFT +
    NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG +
    NEO_TILE_BOTTOM + NEO_TILE_LEFT + NEO_TILE_COLUMNS + NEO_TILE_PROGRESSIVE);

#else
#define mw 16
#define mh 16
#define NUMMATRIX (mw*mh)
CRGB leds[NUMMATRIX];
// Define matrix width and height.
FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, mw, mh,
  NEO_MATRIX_BOTTOM     + NEO_MATRIX_LEFT +
    NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG);
#endif

#define LED_BLACK              0

#define LED_RED_VERYLOW       (3 <<  11)
#define LED_RED_LOW           (7 <<  11)
#define LED_RED_MEDIUM                (15 << 11)
#define LED_RED_HIGH          (31 << 11)

#define LED_GREEN_VERYLOW     (1 <<  5)
#define LED_GREEN_LOW         (15 << 5)
#define LED_GREEN_MEDIUM      (31 << 5)
#define LED_GREEN_HIGH                (63 << 5)

#define LED_BLUE_VERYLOW      3
#define LED_BLUE_LOW          7
#define LED_BLUE_MEDIUM       15
#define LED_BLUE_HIGH         31
```

```
 75
 76 #define LED_ORANGE_VERYLOW     (LED_RED_VERYLOW + LED_GREEN_VERYLOW)
 77 #define LED_ORANGE_LOW         (LED_RED_LOW     + LED_GREEN_LOW)
 78 #define LED_ORANGE_MEDIUM      (LED_RED_MEDIUM  + LED_GREEN_MEDIUM)
 79 #define LED_ORANGE_HIGH               (LED_RED_HIGH    + LED_GREEN_HIGH)
 80
 81 #define LED_PURPLE_VERYLOW     (LED_RED_VERYLOW + LED_BLUE_VERYLOW)
 82 #define LED_PURPLE_LOW         (LED_RED_LOW     + LED_BLUE_LOW)
 83 #define LED_PURPLE_MEDIUM      (LED_RED_MEDIUM  + LED_BLUE_MEDIUM)
 84 #define LED_PURPLE_HIGH               (LED_RED_HIGH    + LED_BLUE_HIGH)
 85
 86 #define LED_CYAN_VERYLOW       (LED_GREEN_VERYLOW + LED_BLUE_VERYLOW)
 87 #define LED_CYAN_LOW           (LED_GREEN_LOW     + LED_BLUE_LOW)
 88 #define LED_CYAN_MEDIUM               (LED_GREEN_MEDIUM  + LED_BLUE_MEDIUM)
 89 #define LED_CYAN_HIGH          (LED_GREEN_HIGH    + LED_BLUE_HIGH)
 90
 91 #define LED_WHITE_VERYLOW      (LED_RED_VERYLOW + LED_GREEN_VERYLOW + LED_BLUE_VERYLOW)
 92 #define LED_WHITE_LOW          (LED_RED_LOW     + LED_GREEN_LOW     + LED_BLUE_LOW)
 93 #define LED_WHITE_MEDIUM       (LED_RED_MEDIUM  + LED_GREEN_MEDIUM  + LED_BLUE_MEDIUM)
 94 #define LED_WHITE_HIGH         (LED_RED_HIGH    + LED_GREEN_HIGH    + LED_BLUE_HIGH)
 95
 96 static const uint16_t PROGMEM
 97       RGB_bmp[][64] = {
 98       // 00: blue, blue/red, red, red/green, green, green/blue, blue, white
 99       { 0x100, 0x200, 0x300, 0x400, 0x600, 0x800, 0xA00, 0xF00,
100         0x101, 0x202, 0x303, 0x404, 0x606, 0x808, 0xA0A, 0xF0F,
101         0x001, 0x002, 0x003, 0x004, 0x006, 0x008, 0x00A, 0x00F,
102         0x011, 0x022, 0x033, 0x044, 0x066, 0x088, 0x0AA, 0x0FF,
103         0x010, 0x020, 0x030, 0x040, 0x060, 0x080, 0x0A0, 0x0F0,
104         0x110, 0x220, 0x330, 0x440, 0x660, 0x880, 0xAA0, 0xFF0,
105         0x100, 0x200, 0x300, 0x400, 0x600, 0x800, 0xA00, 0xF00,
106         0x111, 0x222, 0x333, 0x444, 0x666, 0x888, 0xAAA, 0xFFF, },
107
108       // 01: grey to white
109       { 0x111, 0x222, 0x333, 0x555, 0x777, 0x999, 0xAAA, 0xFFF,
110         0x222, 0x222, 0x333, 0x555, 0x777, 0x999, 0xAAA, 0xFFF,
111         0x333, 0x333, 0x333, 0x555, 0x777, 0x999, 0xAAA, 0xFFF,
112         0x555, 0x555, 0x555, 0x555, 0x777, 0x999, 0xAAA, 0xFFF,
113         0x777, 0x777, 0x777, 0x777, 0x777, 0x999, 0xAAA, 0xFFF,
```

```
114         0x999, 0x999, 0x999, 0x999, 0x999, 0x999, 0xAAA, 0xFFF,
115         0xAAA, 0xAAA, 0xAAA, 0xAAA, 0xAAA, 0xAAA, 0xAAA, 0xFFF,
116         0xFFF, 0xFFF, 0xFFF, 0xFFF, 0xFFF, 0xFFF, 0xFFF, 0xFFF, },
117
118       // 02: low red to high red
119       { 0x001, 0x002, 0x003, 0x005, 0x007, 0x009, 0x00A, 0x00F,
120         0x002, 0x002, 0x003, 0x005, 0x007, 0x009, 0x00A, 0x00F,
121         0x003, 0x003, 0x003, 0x005, 0x007, 0x009, 0x00A, 0x00F,
122         0x005, 0x005, 0x005, 0x005, 0x007, 0x009, 0x00A, 0x00F,
123         0x007, 0x007, 0x007, 0x007, 0x007, 0x009, 0x00A, 0x00F,
124         0x009, 0x009, 0x009, 0x009, 0x009, 0x009, 0x00A, 0x00F,
125         0x00A, 0x00A, 0x00A, 0x00A, 0x00A, 0x00A, 0x00A, 0x00F,
126         0x00F, 0x00F, 0x00F, 0x00F, 0x00F, 0x00F, 0x00F, 0x00F, },
127
128       // 03: low green to high green
129       { 0x010, 0x020, 0x030, 0x050, 0x070, 0x090, 0x0A0, 0x0F0,
130         0x020, 0x020, 0x030, 0x050, 0x070, 0x090, 0x0A0, 0x0F0,
131         0x030, 0x030, 0x030, 0x050, 0x070, 0x090, 0x0A0, 0x0F0,
132         0x050, 0x050, 0x050, 0x050, 0x070, 0x090, 0x0A0, 0x0F0,
133         0x070, 0x070, 0x070, 0x070, 0x070, 0x090, 0x0A0, 0x0F0,
134         0x090, 0x090, 0x090, 0x090, 0x090, 0x090, 0x0A0, 0x0F0,
135         0x0A0, 0x0A0, 0x0A0, 0x0A0, 0x0A0, 0x0A0, 0x0A0, 0x0F0,
136         0x0F0, 0x0F0, 0x0F0, 0x0F0, 0x0F0, 0x0F0, 0x0F0, 0x0F0, },
137
138       // 04: low blue to high blue
139       { 0x100, 0x200, 0x300, 0x500, 0x700, 0x900, 0xA00, 0xF00,
140         0x200, 0x200, 0x300, 0x500, 0x700, 0x900, 0xA00, 0xF00,
141         0x300, 0x300, 0x300, 0x500, 0x700, 0x900, 0xA00, 0xF00,
142         0x500, 0x500, 0x500, 0x500, 0x700, 0x900, 0xA00, 0xF00,
143         0x700, 0x700, 0x700, 0x700, 0x700, 0x900, 0xA00, 0xF00,
144         0x900, 0x900, 0x900, 0x900, 0x900, 0x900, 0xA00, 0xF00,
145         0xA00, 0xA00, 0xA00, 0xA00, 0xA00, 0xA00, 0xA00, 0xF00,
146         0xF00, 0xF00, 0xF00, 0xF00, 0xF00, 0xF00, 0xF00, 0xF00, },
147
148       // 05: 1 black, 2R, 2O, 2G, 1B with 4 blue lines rising right
149       { 0x000, 0x200, 0x000, 0x400, 0x000, 0x800, 0x000, 0xF00,
150         0x000, 0x201, 0x002, 0x403, 0x004, 0x805, 0x006, 0xF07,
151         0x008, 0x209, 0x00A, 0x40B, 0x00C, 0x80D, 0x00E, 0xF0F,
152         0x000, 0x211, 0x022, 0x433, 0x044, 0x855, 0x066, 0xF77,
```

```
153         0x088, 0x299, 0x0AA, 0x4BB, 0x0CC, 0x8DD, 0x0EE, 0xFFF,
154         0x000, 0x210, 0x020, 0x430, 0x040, 0x850, 0x060, 0xF70,
155         0x080, 0x290, 0x0A0, 0x4B0, 0x0C0, 0x8D0, 0x0E0, 0xFF0,
156         0x000, 0x200, 0x000, 0x500, 0x000, 0x800, 0x000, 0xF00, },
157
158       // 06: 4 lines of increasing red and then green
159       { 0x000, 0x000, 0x001, 0x001, 0x002, 0x002, 0x003, 0x003,
160         0x004, 0x004, 0x005, 0x005, 0x006, 0x006, 0x007, 0x007,
161         0x008, 0x008, 0x009, 0x009, 0x00A, 0x00A, 0x00B, 0x00B,
162         0x00C, 0x00C, 0x00D, 0x00D, 0x00E, 0x00E, 0x00F, 0x00F,
163         0x000, 0x000, 0x010, 0x010, 0x020, 0x020, 0x030, 0x030,
164         0x040, 0x040, 0x050, 0x050, 0x060, 0x060, 0x070, 0x070,
165         0x080, 0x080, 0x090, 0x090, 0x0A0, 0x0A0, 0x0B0, 0x0B0,
166         0x0C0, 0x0C0, 0x0D0, 0x0D0, 0x0E0, 0x0E0, 0x0F0, 0x0F0, },
167
168       // 07: 4 lines of increasing red and then blue
169       { 0x000, 0x000, 0x001, 0x001, 0x002, 0x002, 0x003, 0x003,
170         0x004, 0x004, 0x005, 0x005, 0x006, 0x006, 0x007, 0x007,
171         0x008, 0x008, 0x009, 0x009, 0x00A, 0x00A, 0x00B, 0x00B,
172         0x00C, 0x00C, 0x00D, 0x00D, 0x00E, 0x00E, 0x00F, 0x00F,
173         0x000, 0x000, 0x100, 0x100, 0x200, 0x200, 0x300, 0x300,
174         0x400, 0x400, 0x500, 0x500, 0x600, 0x600, 0x700, 0x700,
175         0x800, 0x800, 0x900, 0x900, 0xA00, 0xA00, 0xB00, 0xB00,
176         0xC00, 0xC00, 0xD00, 0xD00, 0xE00, 0xE00, 0xF00, 0xF00, },
177
178       // 08: criss cross of green and red with diagonal blue.
179       { 0xF00, 0x001, 0x003, 0x005, 0x007, 0x00A, 0x00F, 0x000,
180         0x020, 0xF21, 0x023, 0x025, 0x027, 0x02A, 0x02F, 0x020,
181         0x040, 0x041, 0xF43, 0x045, 0x047, 0x04A, 0x04F, 0x040,
182         0x060, 0x061, 0x063, 0xF65, 0x067, 0x06A, 0x06F, 0x060,
183         0x080, 0x081, 0x083, 0x085, 0xF87, 0x08A, 0x08F, 0x080,
184         0x0A0, 0x0A1, 0x0A3, 0x0A5, 0x0A7, 0xFAA, 0x0AF, 0x0A0,
185         0x0F0, 0x0F1, 0x0F3, 0x0F5, 0x0F7, 0x0FA, 0xFFF, 0x0F0,
186         0x000, 0x001, 0x003, 0x005, 0x007, 0x00A, 0x00F, 0xF00, },
187
188       // 09: 2 lines of green, 2 red, 2 orange, 2 green
189       { 0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
190         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
191         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
```

```
192         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
193         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
194         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
195         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0,
196         0x0F0, 0x0F0, 0x0FF, 0x0FF, 0x00F, 0x00F, 0x0F0, 0x0F0, },
197
198       // 10: multicolor smiley face
199       { 0x000, 0x000, 0x00F, 0x00F, 0x00F, 0x00F, 0x000, 0x000,
200         0x000, 0x00F, 0x000, 0x000, 0x000, 0x000, 0x00F, 0x000,
201         0x00F, 0x000, 0xF00, 0x000, 0x000, 0xF00, 0x000, 0x00F,
202         0x00F, 0x000, 0x000, 0x000, 0x000, 0x000, 0x000, 0x00F,
203         0x00F, 0x000, 0x0F0, 0x000, 0x000, 0x0F0, 0x000, 0x00F,
204         0x00F, 0x000, 0x000, 0x0F4, 0x0F3, 0x000, 0x000, 0x00F,
205         0x000, 0x00F, 0x000, 0x000, 0x000, 0x000, 0x00F, 0x000,
206         0x000, 0x000, 0x00F, 0x00F, 0x00F, 0x00F, 0x000, 0x000, },
207 };
208
209 void fixdrawRGBBitmap(int16_t x, int16_t y, const uint16_t *bitmap, int16_t w, int16_t h) {
210     uint16_t RGB_bmp_fixed[w * h];
211     for (uint16_t pixel=0; pixel<w*h; pixel++) {
212         uint8_t r,g,b;
213         uint16_t color = pgm_read_word(bitmap + pixel);
214
215         b = (color & 0xF00) >> 8;
216         g = (color & 0x0F0) >> 4;
217         r = color & 0x00F;
218
219         b = map(b, 0, 15, 0, 31);
220         g = map(g, 0, 15, 0, 63);
221         r = map(r, 0, 15, 0, 31);
222
223         RGB_bmp_fixed[pixel] = (r << 11) + (g << 5) + b;
224             }
225     matrix->drawRGBBitmap(x, y, RGB_bmp_fixed, w, h);
226 }
227
228 // Texto a proyectar en la pantalla
229 void display_scrollText() {
230     uint8_t size = max(int(mh/8), 1);
```

```
231     matrix->clear();
232     matrix->setTextWrap(false);
233     matrix->setTextSize(2);
234     matrix->setRotation(0);
235     for (int8_t x=64; x>=-112; x--) {
236         yield();
237         matrix->clear();
238         matrix->setCursor(x,7);
239         matrix->setTextColor(LED_RED_MEDIUM);
240         matrix->print("ENERGIA");
241         if (mh>11) {
242             matrix->setCursor(x,mh-21);
243       //matrix->setCursor(-64-x,mh-21 );
244             matrix->setTextColor(LED_BLUE_HIGH);
245             matrix->print("UNL");
246         }
247         matrix->show();
248        delay(50);
249     }
250 
251     //Para rotar texto en la pantalla;
252     matrix->setRotation(0);
253     matrix->setTextSize(size);
254     matrix->setTextColor(LED_ORANGE_HIGH);
255     for (int16_t x=8*size; x>=-6*8*size; x--) {
256         yield();
257         matrix->clear();
258         matrix->setCursor(x,mh/2-size*4);
259         matrix->print("ENERGIA");
260         matrix->show();
261         delay(10);
262     }
263     matrix->setRotation(0);
264     matrix->setCursor(0,0);
265     matrix->show();
266 }
267 
268 void display_panOrBounceBitmap (uint8_t bitmapSize) {
269     int16_t xf = max(0, (mw-bitmapSize)/2) << 4;
```

```
270     int16_t yf = max(0, (mh-bitmapSize)/2) << 4;
271     int16_t xfc = 6;
272     int16_t yfc = 3;
273     int16_t xfdir = -1;
274     int16_t yfdir = -1;
275 
276     for (uint16_t i=1; i<200; i++) {
277         bool updDir = false;
278 
279         int16_t x = xf >> 4;
280         int16_t y = yf >> 4;
281 
282         matrix->clear();
283                 // Mapa de pixeles 48x48pixeles
284         if (bitmapSize == 48) matrix->drawRGBBitmap(x, y, (const uint16_t *) bitmap48, bitmapSize, bitmapSize);
285 #ifdef BM32
286         if (bitmapSize == 32) matrix->drawRGBBitmap(x, y, (const uint16_t *) bitmap32, bitmapSize, bitmapSize);
287 #endif
288         matrix->show();
289 
290         if (bitmapSize-mw>2) {
291             xf += xfc*xfdir;
292             if (xf >= 0)                        { xfdir = -1; updDir = true ; };
293             if (xf <= ((mw-bitmapSize) << 4)) { xfdir = 1;  updDir = true ; };
294         }
295         if (bitmapSize-mh>2) {
296             yf += yfc*yfdir;
297             if (yf >= 0)                        { yfdir = -1; updDir = true ; };
298             if (yf <= ((mh-bitmapSize) << 4)) { yfdir = 1;  updDir = true ; };
299         }
300         if (mw>bitmapSize) {
301             xf += xfc*xfdir;
302             if (xf >= (mw-bitmapSize) << 4) { xfdir = -1; updDir = true ; };
303             if (xf <= 0)           { xfdir =  1; updDir = true ; };
304         }
305         if (mh>bitmapSize) {
306             yf += yfc*yfdir;
307             if (yf >= (mh-bitmapSize) << 4) { yfdir = -1; updDir = true ; };
308             if (yf <= 0)           { yfdir =  1; updDir = true ; };
```

```
309         }
310
311         if (updDir) {
312             xfc = constrain(xfc + random(-1, 2), 3, 16);
313             yfc = constrain(xfc + random(-1, 2), 3, 16);
314         }
315     }
316 }
317
318
319 void loop() {
320     static uint8_t pixmap_count = ((mw+7)/8) * ((mh+7)/8);
321 #if 0
322     for (uint8_t i=0; i<100; i++) {
323         matrix->fillScreen(LED_BLUE_LOW);
324         matrix->show();
325         matrix->fillScreen(LED_RED_LOW);
326         matrix->show();
327     }
328 #endif
329
330     Serial.print("Screen pixmap capacity: ");
331     Serial.println(pixmap_count);
332     display_scrollText();
333
334 #ifdef BM32
335         display_panOrBounceBitmap(32);
336 #endif
337         display_panOrBounceBitmap(48);
338        display_panOrBounceBitmap(8);
339 }
340
341 void setup() {
342     delay(10);
343     Serial.begin(115200);
344     Serial.print("Init on pin: ");
345     Serial.println(PIN);
346     Serial.print("Matrix Size: ");
347     Serial.print(mw);
```

```
    Serial.print(" ");
    Serial.print(mh);
    Serial.print(" ");
    Serial.println(NUMMATRIX);

#if defined(P32BY8X3)
    FastLED.addLeds<WS2811_PORTA,3>(leds, NUMMATRIX/3).setCorrection(TypicalLEDStrip);
    Serial.print("Setup parrallel WS2811_PORTA: ");
    Serial.print(NUMMATRIX);
#else
    FastLED.addLeds<NEOPIXEL,PIN>(  leds, NUMMATRIX  ).setCorrection(TypicalLEDStrip);
    Serial.print("Setup serial: ");
    Serial.println(NUMMATRIX);
#endif

    matrix->begin();
    matrix->setTextWrap(false);
    matrix->setBrightness(BRIGHTNESS);
    Serial.println("If the code crashes here, decrease the brightness or turn off the all white display below");
```

Anexo 7. Código de Esp32 Wroom 32s

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <Wire.h>
#include <FastLED.h>
#include "SSD1306Wire.h"
#include "FS.h"
#include "SD.h"
#include "SPI.h"
#include <stdlib.h>


#define PixH 4 * 16
#define PixW 3 * 16
#define NumLEDs (PixH * PixW) + 1
#define PixWxPixH PixW * PixH

String Release = "Panel48x64";
String Date = "10/02/2024";
#define BUILD true

const int I2C_DISPLAY_ADDRESS = 0x3c;
const int SDA_PIN = 21;
const int SCL_PIN = 22;

SSD1306Wire display(I2C_DISPLAY_ADDRESS, SDA_PIN, SCL_PIN);

TaskHandle_t Core0;

#define ACK 6
#define CR 13
#define Debounce 5
#define DEL 127
#define LED_Pin 4
#define LEDSwMax 255
#define LF 10
#define sw0Pin 13
#define swAPin 27
```

```
38 #define swBPin 26
39 #define TESTmode false
40 #define timeout 500
41 #define Xmax PixW-1
42 #define Xmin 0
43 #define Ymax PixH-1
44 #define Ymin 0
45
46 CRGB LED[NumLEDs];
47 CRGB Forgnd[NumLEDs];
48 CRGB Bckgnd[NumLEDs];
49
50 CRGB Sprite0[257];
51 CRGB Sprite1[257];
52 CRGB Sprite2[257];
53 CRGB Sprite3[257];
54 CRGB Sprite4[257];
55 CRGB Sprite5[257];
56 CRGB Sprite6[257];
57 CRGB Sprite7[257];
58
59 CRGB BakRGB[1];
60 CRGB Block[128];
61 CRGB ColRGB[1];
62 CRGB MemRGB[1];
63
64 uint8_t cardType;
65 bool ErrSD;
66 bool SDin = false;
67
68 int16_t ACK_Cnt;
69 bool AnimBuild;
70 int16_t AnimDel;
71 int16_t AnimDp;
72 int16_t AnimErr;
73 int32_t AnimFrame;
74 bool AnimIf;
75 int16_t AnimInc;
76 int16_t AnimLast;
```

```
 77 int16_t AnimLoop;
 78 int16_t AnimMode;
 79 bool AnimPacket;
 80 bool AnimPAUSE;
 81 bool AnimPLAY;
 82 int16_t AnimPnt;
 83 bool AnimSTEP;
 84 bool AnimSTOP;
 85 unsigned long AnimWait;
 86 bool AnyErr;
 87 int AnyInt;
 88 String AnyStr = "";
 89 uint8_t BAK_Blu;
 90 uint8_t BAK_Grn;
 91 uint8_t BAK_Red;
 92 int32_t BckGndFrame;
 93 bool BootAnim;
 94 bool Border;
 95 uint8_t Bright;
 96 bool BrightAuto;
 97 uint8_t BrightCnt;
 98 uint8_t BrightDef;
 99 uint8_t BrightLast;
100 uint8_t Brightness;
101 uint8_t BrightNew;
102 char cmdMode;
103 String cmdRx;
104 int16_t cmdSgn;
105 char cmdType;
106 int16_t cmdVal;
107 uint8_t Col11;
108 uint8_t Col34;
109 uint8_t Col12;
110 uint8_t Col14;
111 byte Col_Pnt;
112 int16_t DELcnt;
113 bool DispBack;
114 bool DispChng;
115 bool DispClr;
```

```
116 int16_t DispCnt;
117 byte DispCX,DispCY;
118 int16_t DispDel;
119 bool DispDisp;
120 bool DispLock;
121 byte DispMenu;
122 int16_t DispMode;
123 bool DispNOP;
124 int16_t DispNow;
125 bool DispOnce;
126 int16_t DispPnt;
127 int16_t DispSub;
128 bool DispSW0;
129 int16_t DispTask;
130 bool DispYN;
131 bool ErrFlg;
132 bool ESC0;
133 String Ext;
134 String Folder;
135 int16_t ForAdd[10];
136 int16_t ForCnt[10];
137 int32_t ForGndFrame;
138 int16_t ForPnt;
139 bool FrameBuild;
140 int16_t FrameCnt;
141 int16_t FrameDel;
142 String FrameMode;
143 int16_t FrameNext;
144 int16_t FrameNum;
145 unsigned long FramePeriod;
146 unsigned long FramePeriodDef;
147 unsigned long FramePeriodLast;
148 bool FrameRESET;
149 int8_t FrameRnd;
150 int16_t FrameSubTask;
151 int16_t FrameTask;
152 unsigned long FrameTimer;
153 String FrameTitle;
154 byte GB[8];
```

```
byte GC[8];
uint8_t Gfx_Char;
int16_t GfxMax;
int16_t GfxP0;
byte GH,GW;
int16_t Gosub[10];
int16_t GosubPnt;
String Hex;
int16_t IcMax;
bool IcOnce;
char keyChar;
int16_t keyVal;
bool LED_Blink;
uint8_t LED_Char;
uint8_t LED_Col;
int16_t LED_Cnt;
int16_t LED_Del;
uint8_t LED_End;
//int8_t LED_Face;    // cube face 0 - 4
bool LED_Flag;
int16_t LED_Inc;
int16_t LED_Last;
uint8_t LED_Max;
uint8_t LED_Min;
bool LEDnoshow;
bool LED_Nop;
int16_t LED_Num;
bool LED_ON;
int16_t LED_Pnt;
int16_t LED_PXY;
uint8_t LED_Rnd;
bool LEDshow;
int16_t LED_SubTask;
bool LED_SwDwn;
int16_t LED_Task;
String LED_Text;
byte LED_Val;
uint8_t LED_X;
uint8_t LED_Y;
```

```
194 int LevelCnt;
195 int LevelOld;
196 unsigned long loopDelay;
197 uint8_t Mem_Blu;
198 uint8_t Mem_Grn;
199 uint8_t Mem_Red;
200 uint8_t MenuLevel;
201 uint8_t MenuPnt;
202 uint8_t MenuT0;
203 uint8_t MenuT1;
204 uint8_t MenuTL;
205 uint8_t MenuTS;
206 String Name;
207 unsigned long next5ms;
208 unsigned long next10ms;
209 unsigned long next40ms;
210 unsigned long PauseTimer;
211 uint8_t Pen_Blu;
212 uint8_t Pen_Grn;
213 uint8_t Pen_Red;
214 bool PrintEn;
215 bool REBOOT;
216 bool RepON;
217 byte ResetCnt;
218 uint8_t RndCol;
219 int16_t RotCnt = 0;
220 int16_t RotEncCnt;
221 int16_t RotState = 0;
222 bool RotUpd;
223 int16_t SchCnt;
224 int16_t SchPnt;
225 int16_t ScrollTask;
226 int16_t SpriteNum;
227 int16_t SpritePnt;
228 int16_t sw0Cnt;
229 int16_t sw0DwnTime;
230 bool sw0LastState;
231 bool SW0_Nop;
232 bool sw0State;
```

```
233 int16_t sw0Timer;
234 bool swAState;
235 bool swBState;
236 unsigned long t0,t1;
237 int16_t Task10ms;
238 int16_t Task40ms;
239 bool TEST;
240 String Text;
241 int16_t Text_SY;
242 int16_t Text_X;
243 int16_t Text_Y;
244 uint8_t Txt_Blu;
245 uint8_t Txt_Char;
246 int16_t Txt_Cnt;
247 uint8_t Txt_Grn;
248 int16_t Txt_Num;
249 int16_t Txt_Pnt;
250 uint8_t Txt_Red;
251 int16_t UsbActive;
252 bool UsbState;
253 bool Verbose;
254 bool WaitUp;
255 String Word;
256 int16_t WordNo;
257 String WordRht;
258 int16_t WordSp;
259
260 // create animation arrays
261 const int AnimDepth = 1000;
262 uint16_t AnimMax = AnimDepth;
263 const int16_t SchDepth = 100;
264
265 uint16_t AnCmd[AnimDepth];
266 int32_t AnD0[AnimDepth];
267 int32_t AnD1[AnimDepth];
268 int32_t AnD2[AnimDepth];
269 String  AnStr[AnimDepth];
270
271 int16_t AnimSch[SchDepth];
```

```
272
273 int16_t SpriteDiv[8] = {1,1,1,1,1,1,1,1};
274 int16_t SpriteFN[8] = {-1,-1,-1,-1,-1,-1,-1,-1};
275 int16_t SpriteInc[8] = {0,0,0,0,0,0,0,0};
276 bool SpriteON[8] = {false,false,false,false,false,false,false,false};
277 int16_t SpriteSX[8] = {0,0,0,0,0,0,0,0};
278 int16_t SpriteSY[8] = {0,0,0,0,0,0,0,0};
279 int16_t SpriteTX[8] = {0,0,0,0,0,0,0,0};
280 int16_t SpriteTY[8] = {0,0,0,0,0,0,0,0};
281 int16_t SpriteX[8] = {0,0,0,0,0,0,0,0};
282 int16_t SpriteY[8] = {0,0,0,0,0,0,0,0};
283
284 int16_t Var[26];
285
286 // ----------------------------------------------------------------------------
287
288 void setup() {
289   // Setup code, runs once:
290   pinMode(sw0Pin,INPUT_PULLUP);
291   pinMode(swAPin,INPUT_PULLUP);
292   pinMode(swBPin,INPUT_PULLUP);
293
294   setDefaults();
295
296   Serial.begin(500000);
297
298   FastLED.addLeds<WS2812B, LED_Pin, GRB>(LED, NumLEDs);
299   FastLED.setBrightness(BrightDef);
300
301   RunPOST();
302
303   synchLoopTimers();
304
305   xTaskCreatePinnedToCore(
306     loop0,
307     "Core0",
308     10000,
309     NULL,
310     1;
```

```
311     &Core0,
312     0);
313 }
314
315 // ------------------------------------------------------------------------------
316 void(* resetFunc) (void) = 0;
317
318 // ------------------------------------------------------------------------------
319
320 void loop() {
321
322   if (AnimBuild) {
323     AnimEngine();
324     if (!AnimBuild) {
325       IcOnce = true;
326       if (AnimMode == 1) {
327         for (int zSP = 0;zSP < 8;zSP++) {
328           if (SpriteON[zSP]) {
329             Gfx_SpriteLoad(zSP);
330           }
331         }
332       }
333     }
334   } else if (FrameBuild) {
335     doFrameTasks();
336   }
337   else if (AnimPLAY && IcOnce) {
338       IcOnce = false;
339     LevelCnt = 0;
340     if (IcMax > 0) {
341       MemRGB[0].setRGB(0,0,0);
342       uint8_t zBN = BrightNew;
343       for (int zL = 1; zL < NumLEDs;zL++) {
344         if (LED[zL] != MemRGB[0]) {
345           LevelCnt+= LED[zL].r; LevelCnt+= LED[zL].g; LevelCnt+= LED[zL].b;
346         }
347       }
348       if (AnimPLAY && (LevelOld != LevelCnt)) {
349
```

```
        float zIb = (0.02 * BrightNew * LevelCnt)/65025.0;
        float zIc = (0.02 * LevelCnt)/255.0;
        float zIcMax = 0.1 * IcMax;
        if (zIc > zIcMax) {
          Bright = (zIcMax * 255)/zIc;
          if (Bright < BrightNew) {BrightNew = Bright; FastLED.setBrightness(BrightNew); BrightCnt = 0;}
        }
        if (Verbose) {Serial.println(String(LevelCnt) + "   " + String(zIc) + " Amps   " + String(BrightNew) + "   " +
String((0.02 * BrightNew * LevelCnt)/65025) + " Amps");}
      }
      LevelOld = LevelCnt;
      if (BrightNew > 0) {
        int16_t zLED0 = 1 + (255/BrightNew);
        if (BrightNew < zBN) {LED[0].setRGB(zLED0,0,0);}
        else {LED[0].setRGB(0,zLED0,0);}
      }
    }
  }

  else if ((millis() - FrameTimer) >= FramePeriod) {
    FrameTimer = millis();
    if (LEDnoshow) {
              LEDshow = false; LEDnoshow = false;
    }
    if (AnimDel > 0) {AnimDel--;}
    else {

      if (LEDshow) {
        LEDshow = false;
        if (AnimSTEP && !AnimPAUSE) {AnimSTEP = false; AnimPAUSE = true;}
        FastLED.show();
      }

      if (AnimMode == 0) {
        FrameBuild = true;
      } else if (AnimMode == 1) {
            if (AnimPLAY && !AnimPAUSE) {AnimBuild = true;}
      }
    }
```

```
  }
  if (AnimSTOP) {
    Serial.println("AnimSTOP");
    ResetAnimEngines(); AnimSTOP = false;
  }

  if (Task40ms > 0) {
    switch(Task40ms) {
      case  1: Display_40ms(); break;
    } Task40ms--;
  } else if ((millis() - next40ms) >= 40) {
    next40ms = millis(); Task40ms = 1;
  }
}

void loop0(void * pvParameters) {

  Serial.print("Core0 task running on core "); Serial.println(xPortGetCoreID());

  for(;;){

      while (Serial.available() > 0) {
      readSerial();
    }

       if (!UsbState && BUILD && ((millis() - next5ms) >= Debounce)) {
      next5ms = millis();
      swAState = digitalRead(swAPin); swBState = digitalRead(swBPin);
      switch (RotState) {
        case -1:
          if (swAState && swBState) {RotState = 0; RotCnt= 0;}
          break;
        case 0:
                if ((!swAState) && (!swBState)) {RotState = -1;}
          else if (!swAState) {RotState = 1;}
          else if (!swBState) {RotState = 4;}
          break;
        case 1:

```

```
428           if (!swBState) {RotState = 2;}
429           else if (swAState) {RotState = 0; RotCnt= 0;}
430           break;
431         case 2:
432           if (swAState) {RotState = 3;}
433           else if (swBState) {RotState = 1;}
434           break;
435         case 3:
436           if (swBState) {RotState = 0; RotCnt = 0; RotEncCnt++; RotUpd = true;}
437           else if (!swAState) {RotState = 2;}
438           break;
439         case 4:
440           if (!swAState) {RotState = 5;}
441           else if (swBState) {RotState = 0; RotCnt= 0;}
442           break;
443         case 5:
444           if (swBState) {RotState = 6;}
445           else if (swAState) {RotState = 4;}
446           break;
447         case 6:
448           if (swAState) {RotState = 0; RotCnt = 0; RotEncCnt--; RotUpd = true;}
449           else if (!swBState) {RotState = 5;}
450           break;
451       }
452       if (RotState > 0) {RotCnt++; if (RotCnt > timeout) {RotState = -1;}}
453     }
454 
455     if (Task10ms > 0) {
456       switch(Task10ms) {
457         case 1: if (RotUpd) {MenuCheck();}; break;
458         case 2: if (BUILD) {readSW0();} break;
459       } Task10ms--;
460     } else {
461       if ((millis() - next10ms) >= 10) {
462         next10ms = millis();
463         if (UsbActive > 0) {
464                     UsbActive--;
465           if (keyVal != ACK) {ACK_Cnt = 0;}
466           if ((!UsbState) && (ACK_Cnt == 3)) {
```

```
467             DispLock = true;
468             UsbState = true;
469             AnimSTOP = true;
470             BootAnim = false;
471           }
472           if (UsbActive < 1) {
473             UsbState = false;
474             DispNOP = false;
475           }
476           Task10ms = 0;
477         } else {Task10ms = 2;}
478       }
479     }
480     if ((millis() - loopDelay) > 10) {
481       loopDelay = millis();
482       vTaskDelay(1);
483     }
484   }
485 }
486
487 // ----------------------------------------------------------------------------------
488
489 void RunPOST() {
490   Serial.println("\n\n" + Release + "   " + Date + "\n");
491   FastLED.clear();
492   FastLED.show();
493   Serial.println("RGB LEDs cleared.");
494
495   if (BUILD) {
496     Display_Init();
497     display.setBrightness(Brightness);
498     Display_Intro();
499     Serial.println("Display initialised");
500     DispDel = 50;
501     DispClr = true;
502   } else {
503     Serial.println("BUILD defines display as disconnected!");
504   }
505
```

```
506     initSD();
507   SDin = ErrSD;
508
509   while (Serial.available() > 0) {keyVal = Serial.read();}
510
511
512   if (!digitalRead(sw0Pin)) {
513
514     TEST = true;
515   }
516
517   BootAnim = false;
518   if (TEST) {
519     Serial.println("TEST mode");
520   } else {
521     Serial.println("Normal mode");
522     MenuSet(0);
523   }
524
525   ClearAnimArray();
526   readSDAnim("0000");
527   if (ErrFlg) {
528     Serial.println("No animation script!");
529   } else {
530     setAnimMode(1); AnimPLAY = true;
531     Serial.println("Playing animation boot file from SD");
532     BootAnim = true;
533     RepON = false;
534     Verbose = false;
535   }
536
537   Serial.println("POST completed");
538 }
539
540 // ------------------------------------------------------------------------------
541
542 void setDefaults() {
543   ACK_Cnt = 0;
544   AnimBuild = false;
```

```
545   AnimDel = 0;
546   AnimErr = 0;
547   AnimFrame = -1;
548   AnimIf = true;
549   AnimMode = -1;
550   AnimPacket = false;
551   AnimPAUSE = false;
552   AnimPLAY = false;
553   AnimPnt = 0;
554   AnimSTEP = false;
555   AnimSTOP = false;
556   AnimWait = 0;
557   BckGndFrame = -1;
558   Border = false;
559   Bright = BrightDef;
560   BrightAuto = true;
561   BrightCnt = 0;
562   BrightDef = 64;
563   Brightness = 192;
564   BrightNew = 255;
565   cmdMode = ' ';
566   cmdSgn = 1;
567   cmdType = ' ';
568   cmdVal = 0;
569   Col11 = 255;
570   Col34 = 192;
571   Col12 = 128;
572   Col14 = 64;
573   DELcnt = 0;
574   DispBack = false;
575   DispChng = false;
576   DispClr = false;
577   DispCnt = 0;
578   DispDel = 0;
579   DispDisp = false;
580   DispLock = false;
581   DispMenu = 0;
582   DispMode = 0;
583   DispNOP = false;
```

```
584   DispNow = 0;
585   DispPnt = 0;
586   DispSub = 0;
587   DispTask = 0;
588   Ext = ".txt";
589   LED_Pnt = 0;
590   LEDnoshow = false;
591   loopDelay = 10;
592   Folder = "";
593   ForGndFrame = -1;
594   ForPnt = -1;
595   FrameBuild = false;
596   FrameCnt = 0;
597   FrameDel = 0;
598   FrameMode = "";
599   FrameNext = 0;
600   FramePeriod = 40;
601   FramePeriodDef = 0;
602   FrameTask = 0;
603   FrameTitle = "";
604   GfxP0 = NumLEDs - 1;
605   GosubPnt = -1;
606   IcMax = 0;
607   IcOnce = false;
608   LED_Blink = false;
609   LED_Cnt = 0;
610   LED_Col = 0;
611   LED_Del = 0;
612   LED_Inc = 1;
613   LED_Max = 32;
614   LED_Nop = false;
615   LED_ON = false;
616   LEDshow = false;
617   LED_SubTask = 0;
618   LED_SwDwn = false;
619   LED_Task = 0;
620   LED_Val = 0;
621   LevelCnt = 0;
622   LevelOld = 0;
```

```
623   Mem_Blu = 0;
624   Mem_Grn = 0;
625   Mem_Red = 0;
626   MenuLevel = 0;
627   MenuPnt = 0;
628   MenuT0 = 0;
629   MenuT1 = 2;
630   Name = "";
631   PrintEn = true;
632   REBOOT = false;
633   RepON = true;
634   ResetCnt = 0;
635   RndCol = 255;
636   RotEncCnt = 32;
637   RotUpd = true;
638   SchCnt = 0;
639   SchPnt = -1;
640   ScrollTask = 0;
641   sw0Cnt = 0;
642   sw0DwnTime = 0;
643   sw0LastState = HIGH;
644   SW0_Nop = false;
645   sw0State = HIGH;
646   sw0Timer = 0;
647   Task10ms = 0;
648   Task40ms = 0;
649   TEST = TESTmode;
650   Text = "";
651   Text_SY = 0;
652   Text_X = 0;
653   Text_Y = 0;
654   Txt_Blu = 255;
655   Txt_Grn = 0;
656   Txt_Red = 0;
657   UsbActive = 0;
658   UsbState = false;
659   Verbose = true;
660   WaitUp = false;
661   Word = "";
```

```
662   WordRht = "";
663   WordSp = -1;
664
665   SerialFlush();
666 }
667
668 // ---------------------------------------------------------------------------
669
670 void synchLoopTimers() {
671   FrameTimer = millis() + 40;
672   next5ms = millis();
673   next10ms = millis() + 1;
674   next40ms = millis() + 2;
```

Anexo 8. Ejemplo de código UTFT para el logo UNL.

```
// Generated by   : ImageConverter 565 Online
// Generated from : F48.jpg
// Time generated : Sun, 18 Feb 24 01:27:18 +0100  (Server timezone: CET)
// Image Size     : 48x48 pixels
// Memory usage   : 4608 bytes


#if defined(__AVR__)
    #include <avr/pgmspace.h>
#elif defined(__PIC32MX__)
    #define PROGMEM
#elif defined(__arm__)
    #define PROGMEM
#endif

const unsigned short bitmap48[2304] PROGMEM={
0x0020, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0800, 0x0820, 0x0820, 0x1040, 0x2920, 0x3140, 0x5A40, 0x7300, 0x6AC0, 0x6AC0, 0x5200, 0x3920, 0x1000, 0x0820, 0x0820, 0x0020,
0x0020, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0001, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000,
0x0020, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0800, 0x0800, 0x0800,
```

```
0x49E1, 0x9C69, 0xCDED, 0xDE6D, 0xD60C, 0xCDA9, 0xC588, 0xBD26, 0xBD67, 0xBD47, 0xC588, 0xD60B, 0xDE2D, 0xACEA, 0x7BA6, 0x2120,
0x0040, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0000, 0x0820, 0x83C4, 0xDE6C,
0xCDC9, 0xB4E5, 0xA463, 0x9C21, 0x9C00, 0xA420, 0xACA1, 0xA480, 0xB522, 0xA4C1, 0xA4A1, 0x9C40, 0xA461, 0xB4C3, 0xC546, 0xDE2A,
0xBD89, 0x6B01, 0x1000, 0x0800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0861, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0021, 0x0000, 0x0000, 0x0020, 0x0020, 0x0020, 0x3160, 0xE6D1, 0xBD66, 0xA460,
0x9420, 0xA460, 0xB524, 0xCDC6, 0xD5C6, 0xE689, 0xEF0B, 0xEF0B, 0xB586, 0xD649, 0xEE4A, 0xF66A, 0xDDC6, 0xCD03, 0xBCA2, 0xAC40,
0xBCE2, 0xC587, 0xD5ED, 0x7B46, 0x0800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0820, 0x3140, 0xD5EB, 0x9C21, 0x9C20, 0xBD62,
0xDEA7, 0xD6A8, 0xDF09, 0xF7AD, 0xD6AA, 0xDF0C, 0x7421, 0x7CA5, 0x94EA, 0xD611, 0xA346, 0xC3E7, 0xD486, 0xFEAC, 0xFE6B, 0xF5E8,
0xCD43, 0xB4C1, 0xAC83, 0xC568, 0xCDEB, 0x49E0, 0x0820, 0x0020, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0021,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0020, 0x2920, 0xB50A, 0xB507, 0x9C00, 0xBD22, 0xD606, 0xE6E9,
0x73E0, 0xA604, 0xCF2A, 0x7CC1, 0x8503, 0x7CE3, 0x8566, 0x6445, 0xE799, 0xDE37, 0xED74, 0x7900, 0x8900, 0xF3E6, 0x8880, 0xDB83,
0xF527, 0xF5A7, 0xD505, 0xBC82, 0xB4E3, 0xCDE9, 0x8C27, 0x0020, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0820, 0x7B63, 0xDE2B, 0x9C01, 0xB4A3, 0xDE06, 0xE688, 0xFFEF, 0xBEA9,
0x7501, 0x6D83, 0x6522, 0x75A4, 0x7DA4, 0x7DC4, 0x7DA6, 0x6445, 0xF7BB, 0xFF5D, 0xCC50, 0x8902, 0xC0A0, 0xD020, 0xD8A0, 0xC880,
0xB0A0, 0xC221, 0xFE8E, 0xF5C7, 0xBD02, 0xA4C1, 0xBDA6, 0x9CC7, 0x0840, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0820, 0xAD2A, 0xC587, 0x9C00, 0xC523, 0xEEA9, 0xEF2A, 0xDF0A, 0xAE48, 0x6CA2,
0x7565, 0x75C6, 0x75E6, 0x7626, 0x7604, 0x75E4, 0x7DE6, 0x6CA7, 0xC636, 0xDDF8, 0xFE9A, 0x88A2, 0xD862, 0xF800, 0xF820, 0xE820,
0xD840, 0xC080, 0xD241, 0x8940, 0xF628, 0xCDA4, 0xA4A0, 0xB565, 0xBD8B, 0x0820, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x7343, 0xC567, 0xA441, 0xC564, 0xE6A8, 0xCE67, 0x74C1, 0x64C1, 0x7563, 0x7D85,
0x5C64, 0x8E0A, 0x6D65, 0x75E5, 0x7604, 0x7605, 0x75C6, 0x7D6A, 0xFFFD, 0xFFBF, 0xFF3D, 0x8882, 0xE062, 0xF820, 0xE800, 0xF040,
0xE861, 0xD840, 0xD860, 0xD160, 0xF465, 0xFE29, 0xDDE6, 0xB4E2, 0xB525, 0xBD4A, 0x0820, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x4A20, 0xCDC9, 0x9C20, 0xC564, 0xF70A, 0xE76A, 0x8D83, 0x6D42, 0x6DC4, 0x75E4, 0x75A5,
0x7527, 0x9E2D, 0x64A4, 0x7DA6, 0x7DE5, 0x75E5, 0x7DE7, 0x5445, 0xF7FD, 0xFFDF, 0xFF1C, 0x8882, 0xD082, 0xE841, 0xD882, 0xC861,
0xB800, 0xD0A2, 0xE062, 0xC800, 0xA8C0, 0xEC25, 0xE525, 0xDDA5, 0xB4A1, 0xBD46, 0x8BE6, 0x0800, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0020, 0x0000, 0x0020, 0x2920, 0xD62B, 0xA441, 0xC564, 0xDE26, 0xDF09, 0x6440, 0x7583, 0x75C4, 0x6DE4, 0x7605, 0x75C5,
0x7567, 0x7D08, 0xD774, 0x8D4B, 0x64A5, 0x7DC7, 0x7E06, 0x5CC5, 0xC6F6, 0xFFFF, 0xFF3C, 0x9104, 0xC8C3, 0x9800, 0x7820, 0x5000,
0x4800, 0x7841, 0x9800, 0xD882, 0xD060, 0xB080, 0x9940, 0xFDA8, 0xD585, 0xB502, 0xCDA7, 0x5A80, 0x0800, 0x0000, 0x0000, 0x0020,
0x0000, 0x0000, 0x0000, 0x0820, 0xDE6D, 0xACA2, 0xBD03, 0xDE27, 0xFFAC, 0xBEA8, 0x6501, 0x6DC3, 0x7604, 0x7604, 0x75E4, 0x7DC6,
0x7D87, 0x4B42, 0xF7F8, 0xD6F5, 0xBED1, 0x6CE6, 0x75C5, 0x7587, 0x958F, 0xFFFE, 0xFF9C, 0x7923, 0xB0E3, 0x9000, 0x3800, 0x3081,
0x2881, 0x3000, 0x8800, 0xD083, 0xE041, 0xD840, 0xB0A0, 0xF446, 0xFE89, 0xC542, 0xBCE2, 0xD60B, 0x0800, 0x0000, 0x0000, 0x0020,
0x0000, 0x0820, 0x0800, 0xA4C9, 0xB4E5, 0xB4A1, 0xD5C5, 0xFF8C, 0xAE27, 0x8584, 0x6D62, 0x7603, 0x75E4, 0x75E4, 0x75C4, 0x7DA5,
0x74A5, 0xCF11, 0xDF14, 0xFFD9, 0xEF56, 0x84C8, 0x7DA5, 0x7DE6, 0x6C89, 0xFFFD, 0xE6F8, 0xCCD1, 0x9923, 0x8041, 0x932C, 0xCDF7,
0xCE38, 0x9B8D, 0x8000, 0xC882, 0xE040, 0xE840, 0xD080, 0xD201, 0xED06, 0xE606, 0xBD02, 0xBD26, 0x62A2, 0x0800, 0x0000, 0x0020,
0x0000, 0x0000, 0x3120, 0xDE2B, 0xAC81, 0xCD84, 0xDE87, 0xDF2A, 0x5C40, 0x6D64, 0x7E25, 0x7603, 0x7604, 0x75E4, 0x7E05, 0x85E6,
0x6CA5, 0xDF73, 0xBDF0, 0xFFB8, 0xE6D5, 0x8C88, 0x7DA5, 0x7DE5, 0x53A5, 0xDF58, 0xF7DA, 0xB4F0, 0x9227, 0x3800, 0x38E3, 0x39C7,
0x39E7, 0x3082, 0x4000, 0xB903, 0xE040, 0xF000, 0xE020, 0xC080, 0xBA80, 0xF5C7, 0xDDE5, 0xB4E3, 0xC5AB, 0x2900, 0x0000, 0x0000,
0x0000, 0x0820, 0xC58A, 0xACA2, 0xC542, 0xD605, 0xF7CD, 0xA606, 0x7522, 0x7E05, 0x6DE4, 0x6E04, 0x7624, 0x7624, 0x7604, 0x7DE5,
```

```
    0x7D86, 0xA64C, 0xBE0F, 0xFFB8, 0xD673, 0x8488, 0x75A5, 0x75E5, 0x956B, 0xA54F, 0xFFFB, 0xBD73, 0x7B09, 0x1000, 0x2904, 0x526A,
    0x5249, 0x3924, 0x2000, 0x7860, 0xD881, 0xF820, 0xE800, 0xD860, 0xA8A0, 0xFDAA, 0xF647, 0xC583, 0xAD05, 0x7B84, 0x0820, 0x0000,
    0x0000, 0x3140, 0xD5EA, 0xB4A1, 0xD5E4, 0xFFAB, 0xB687, 0x6CE1, 0x7DC3, 0x75E3, 0x7604, 0x7625, 0x6E04, 0x7604, 0x6DE4, 0x7605,
    0x75C6, 0x7D27, 0xCEB3, 0xFFFA, 0xCE73, 0x7487, 0x75E5, 0x75E5, 0x7465, 0xBDCF, 0xC5D2, 0xCDD3, 0x8C2D, 0x0000, 0x2945, 0x4A29,
    0x524A, 0x3124, 0x1000, 0xB30A, 0xC881, 0xF000, 0xF020, 0xE860, 0xC8A0, 0xCA61, 0xFDC7, 0xDDE5, 0xB523, 0x9465, 0x0800, 0x0000,
    0x0800, 0x9447, 0xB525, 0xB501, 0xD646, 0x94E0, 0x64A0, 0x7DE5, 0x7604, 0x7604, 0x7604, 0x7604, 0x7604, 0x7624, 0x7604, 0x75E4,
    0x7DE6, 0x6CE6, 0x4284, 0x0040, 0x744A, 0x64A6, 0x75E5, 0x7E05, 0x6CA4, 0xB62E, 0xF755, 0xFF77, 0x942C, 0x0820, 0x2924, 0x4A49,
    0x526A, 0x2944, 0x0800, 0xFE35, 0xB080, 0xE020, 0xF020, 0xF040, 0xD880, 0xD9E1, 0xFE0A, 0xFE48, 0xBD03, 0xAD07, 0x0800, 0x0000,
    0x1020, 0xCDEB, 0xACA2, 0xC583, 0xEF09, 0xA5C3, 0x7542, 0x7604, 0x7603, 0x7604, 0x7604, 0x7604, 0x7604, 0x6DE4, 0x75E4, 0x7DE5,
    0x7DA6, 0x7508, 0x0080, 0x0040, 0x0960, 0x85C9, 0x7E26, 0x75E4, 0x7525, 0x9DAA, 0xFF53, 0xF6D4, 0xA4CE, 0x0000, 0x2923, 0x4A48,
    0x5269, 0x2923, 0x0800, 0xFE96, 0xB162, 0xD8A0, 0xE840, 0xE800, 0xE000, 0xC880, 0x90C0, 0xF587, 0xD586, 0xBD67, 0x4A00, 0x0820,
    0x3120, 0xCDC9, 0xB4C2, 0xCDC4, 0xDEE8, 0x7460, 0x85A4, 0x7605, 0x7624, 0x7604, 0x6E04, 0x6E04, 0x7604, 0x7625, 0x75E4, 0x7DE6,
    0x6485, 0xF7F8, 0x8C4C, 0xAD90, 0x74A7, 0x85C8, 0x7DE5, 0x7E05, 0x7DA6, 0x6C43, 0xE6B0, 0xFF15, 0x9CAE, 0x0020, 0x2144, 0x4A48,
    0x5AA9, 0x28E1, 0x1000, 0xFE75, 0xCB29, 0xB080, 0xE040, 0xF820, 0xF820, 0xE080, 0xE281, 0xFDCA, 0xEE07, 0xB504, 0x8C27, 0x0820,
    0x6280, 0xC547, 0xBD02, 0xD605, 0xC605, 0x9DA4, 0x6D22, 0x75E4, 0x75E4, 0x7604, 0x7625, 0x6E04, 0x6DE4, 0x6DE4, 0x7605, 0x6D25,
    0xA60E, 0xFFFA, 0xF779, 0x2140, 0x2A60, 0x7DC7, 0x75E5, 0x7DE5, 0x7D86, 0x5381, 0xFF73, 0xF6D4, 0x83AA, 0x0000, 0x2944, 0x41E7,
    0xB533, 0x4183, 0x2060, 0xF613, 0xCBCA, 0xC1E4, 0xD840, 0xF800, 0xF800, 0xE040, 0xB060, 0xD302, 0xFE48, 0xB503, 0x9CA8, 0x0820,
    0x83A0, 0xBD05, 0xC522, 0xDE25, 0xFFEE, 0xC6E9, 0x6D42, 0x7604, 0x7603, 0x7603, 0x7604, 0x7605, 0x7645, 0x7604, 0x75A5, 0x6CC5,
    0xFFF9, 0xFFDA, 0x840C, 0x0060, 0x00A0, 0x3BE0, 0x7626, 0x7E06, 0x6464, 0xCEAF, 0xFF94, 0xDE31, 0x4982, 0x1000, 0x28C3, 0x4A28,
    0x8BEC, 0x93EA, 0x40E0, 0xFEB4, 0xCBEA, 0xB9C2, 0xE060, 0xF800, 0xF800, 0xE840, 0xD8A0, 0xDA62, 0xFDE7, 0xBD43, 0xB549, 0x1040,
    0x8BE0, 0xBD24, 0xC542, 0xF708, 0xBE26, 0x74A1, 0x7583, 0x75E4, 0x7604, 0x7624, 0x7604, 0x7604, 0x6DE4, 0x75C5, 0x6CE4, 0xCF31,
    0xFFF8, 0xEF57, 0xB571, 0x3A43, 0x7CEA, 0x6CE7, 0x8628, 0x5C82, 0xCF10, 0xFFB5, 0xF6D2, 0xB42A, 0x6101, 0x2800, 0x30A3, 0x5249,
    0x5205, 0xFF16, 0xFED4, 0xFE32, 0xDC2B, 0xB161, 0xE060, 0xF800, 0xF000, 0xE800, 0xD820, 0xC140, 0xFE28, 0xCDC4, 0xBD69, 0x1020,
    0x7B60, 0xC564, 0xCD81, 0xE6C6, 0x84C0, 0x7D83, 0x7DE5, 0x75E4, 0x75C4, 0x75E4, 0x75E4, 0x7625, 0x7605, 0x7DA6, 0x854A, 0xFFF9,
    0xF796, 0xEF35, 0xBDF0, 0x6BA8, 0x0080, 0x3262, 0x7CE8, 0xC6CF, 0xF793, 0xFEF2, 0xE52E, 0x8981, 0xA922, 0x6000, 0x7209, 0x7B4D,
    0x6AC8, 0xD591, 0xE5AF, 0xE4CB, 0xDBEA, 0xA8E0, 0xE060, 0xF800, 0xF820, 0xF000, 0xE020, 0xD140, 0xE4E3, 0xCDE4, 0xBD88, 0x1060,
    0xA463, 0xB4E3, 0xD5E2, 0xD684, 0x8521, 0x7584, 0x75E4, 0x7604, 0x7604, 0x7604, 0x7604, 0x7604, 0x7DE5, 0x6CE5, 0xD754, 0xFFD9,
    0xF776, 0xE714, 0xA56C, 0x2160, 0x0060, 0x0040, 0xB5CE, 0xFFB5, 0xEED1, 0xFE51, 0xB265, 0xA8E1, 0xC8C2, 0xA040, 0x91E8, 0x9B8D,
    0x7AE9, 0xBC8D, 0xE56E, 0xCBC7, 0xD2A5, 0xC0A0, 0xE040, 0xF800, 0xF800, 0xF020, 0xE040, 0xB8A0, 0xFDC7, 0xD625, 0xBD67, 0x4A00,
    0xA462, 0xBD23, 0xCDC3, 0xD686, 0xD78C, 0x7563, 0x7605, 0x6DE4, 0x7603, 0x7603, 0x7604, 0x7605, 0x75C5, 0x6CC5, 0xFFF9, 0xF758,
    0xF756, 0xDED2, 0x9509, 0x5BA3, 0x5365, 0x7428, 0xFFF7, 0xEEB2, 0xF6F2, 0xD44A, 0xA040, 0xD882, 0xD881, 0xA000, 0x7000, 0x5000,
    0x3800, 0xEE12, 0xE52D, 0xDBE8, 0xB8E0, 0xD860, 0xF000, 0xF800, 0xF800, 0xE820, 0xD880, 0xFB26, 0xF586, 0xD605, 0xC567, 0x6AC0,
    0xAC42, 0xBCE3, 0xCDA3, 0xF7AA, 0x8522, 0x6D22, 0x7605, 0x6E04, 0x6E03, 0x7644, 0x75E3, 0x75E4, 0x75A6, 0x9E0B, 0xFFF8, 0xFF57,
    0xFF56, 0xD6B2, 0x7CA7, 0x8549, 0x74A8, 0xF7F6, 0xF713, 0xFED2, 0xEE30, 0x89C1, 0xC0E1, 0xE061, 0xD840, 0xD0C1, 0xA901, 0xAA65,
    0xFEB5, 0xFE52, 0xD429, 0xC242, 0xD0A0, 0xE820, 0xF800, 0xF800, 0xF820, 0xE840, 0xD060, 0xCA01, 0xFE28, 0xDE26, 0xBD06, 0x51E0,
    0xA422, 0xBD04, 0xC543, 0xCE04, 0x7CA0, 0x7D83, 0x7605, 0x7604, 0x7624, 0x7623, 0x7603, 0x75E5, 0x7546, 0xBECF, 0xFFD6, 0xF735,
    0xF735, 0xDED3, 0x7CC7, 0x74A6, 0xBE6F, 0xF774, 0xF6D0, 0xFED1, 0xD52C, 0x60A0, 0xB121, 0xC880, 0xE061, 0xC8A0, 0xBAA6, 0xFED4,
    0xFE51, 0xD46A, 0xDBA8, 0xA8A0, 0xD880, 0xF020, 0xF800, 0xF800, 0xF000, 0xE820, 0xD080, 0xA900, 0xFE08, 0xD5C5, 0xBD47, 0x49A0,
    0x8B80, 0xBD04, 0xCD84, 0xD625, 0xADE4, 0x7D41, 0x75A3, 0x7604, 0x7604, 0x7604, 0x7E05, 0x7DC6, 0x6CC6, 0xDF94, 0xFFD6, 0xFF55,
```

```
 99    0xFF76, 0xCE71, 0x7426, 0x7CA7, 0xFFD4, 0xEED1, 0xF6AF, 0xFED0, 0xFE91, 0xFEB5, 0xE2A7, 0xC0A1, 0xC8C1, 0xC9C3, 0xFEB4, 0xFE30,
100    0xDCEB, 0xD3E8, 0xB9A2, 0xC8C0, 0xE860, 0xF800, 0xF800, 0xF800, 0xF800, 0xE840, 0xFA46, 0xFC49, 0xFE08, 0xBD43, 0xB506, 0x6280,
101    0x6AC0, 0xBD25, 0xC522, 0xCE04, 0xDEE7, 0xA5E3, 0x7582, 0x7624, 0x7624, 0x75E4, 0x6524, 0x7528, 0x6427, 0xF7F8, 0xF775, 0xF755,
102    0xF756, 0xBDAF, 0x52C3, 0xD6B1, 0xF711, 0xF6CF, 0xFEF0, 0xF6D0, 0xFEF3, 0xED6F, 0xA040, 0xC0C1, 0x9900, 0xFE74, 0xFDF0, 0xE52C,
103    0xC3E8, 0xDB67, 0xC060, 0xE820, 0xF020, 0xF820, 0xF800, 0xF800, 0xF800, 0xE060, 0xB060, 0xFCC9, 0xFEAA, 0xB522, 0xBD68, 0x5A40,
104    0x3120, 0xC589, 0xB4E2, 0xD625, 0xF749, 0x9D41, 0x7561, 0x7E24, 0x7604, 0x75A4, 0x960A, 0xB6D1, 0x8D2C, 0xFFF9, 0xFF96, 0xFF55,
105    0xF755, 0xB54D, 0xF754, 0xFF94, 0xF6B0, 0xF6AF, 0xF6D0, 0xF6D1, 0xF6F2, 0xBBC8, 0xA0A0, 0xA8A0, 0xFD70, 0xFE31, 0xE54D, 0xD469,
106    0xDC2A, 0xB161, 0xD860, 0xF000, 0xF800, 0xF800, 0xF800, 0xF800, 0xF820, 0xE060, 0xB920, 0xD3C3, 0xFE89, 0xA4A1, 0xD62C, 0x1840,
107    0x0820, 0xB54B, 0xB504, 0xCDC4, 0xE6C7, 0x94C0, 0x7D82, 0x75E3, 0x75C3, 0x7DC5, 0x74C7, 0xF7F8, 0xFFF9, 0xF797, 0xF775, 0xFF75,
108    0xFF76, 0xACEB, 0xFF33, 0xF6F1, 0xF6D0, 0xFED0, 0xF6D0, 0xFED2, 0xFE93, 0x8A43, 0x9120, 0xDB27, 0xFE52, 0xED8E, 0xDCEA, 0xD408,
109    0xD2E6, 0xB8C0, 0xE020, 0xF820, 0xF800, 0xF800, 0xF800, 0xF800, 0xF000, 0xD860, 0xA900, 0xE485, 0xE607, 0xB504, 0xB529, 0x0820,
110    0x0000, 0x20C0, 0xC567, 0xC584, 0xDE25, 0xBDC3, 0x95C3, 0x7582, 0x7E04, 0x7DC5, 0x6CA6, 0xEFF6, 0xEF76, 0xF755, 0xF754, 0xF775,
111    0xF776, 0xB52C, 0xFF12, 0xEEB0, 0xF6D0, 0xF6D0, 0xF6B0, 0xFEF3, 0xEDD1, 0x7160, 0x89C0, 0xFE72, 0xFDCE, 0xDD2B, 0xCC69, 0xDC09,
112    0xB141, 0xC880, 0xF040, 0xF800, 0xF800, 0xF800, 0xF800, 0xF000, 0xF060, 0xE981, 0xFC66, 0xF5A8, 0xC523, 0xB525, 0x83E6, 0x0820,
113    0x0020, 0x0800, 0xACC6, 0xBD23, 0xDE24, 0xF728, 0xAE46, 0x7542, 0x75E3, 0x7E05, 0x6CC4, 0xCF10, 0xF796, 0xFF55, 0xF775, 0xF755,
114    0xDEB4, 0xB52D, 0xEED1, 0xDE2D, 0xDE4E, 0xF6D0, 0xF6F1, 0xD58D, 0xB3EA, 0x58E0, 0xF5D0, 0xFE30, 0xE54B, 0xDD0B, 0xCC29, 0xBAE6,
115    0xB0C0, 0xD881, 0xF020, 0xF800, 0xF800, 0xF800, 0xF820, 0xF000, 0xD880, 0xFBC7, 0xECE5, 0xFEC9, 0xAC60, 0xC588, 0x0840, 0x0020,
116    0x0000, 0x0820, 0x6280, 0xBD25, 0xCDA3, 0xE6A6, 0xA541, 0x8D82, 0x75A3, 0x7605, 0x7565, 0x9589, 0xFFB6, 0xF755, 0xF754, 0xF775,
117    0xB56F, 0xAD0C, 0xEF12, 0xE68E, 0xC58A, 0xF6CF, 0xF6F1, 0xCD4D, 0xEDD1, 0xC46B, 0xFED3, 0xDD8C, 0xCCC9, 0xDCEA, 0xCC29, 0x91A1,
118    0xDA05, 0xD080, 0xE820, 0xF820, 0xF800, 0xF800, 0xF800, 0xE840, 0xC8A0, 0xB1C0, 0xFE8A, 0xC563, 0xBD24, 0x7B41, 0x0020, 0x0020,
119    0x0000, 0x0000, 0x1860, 0xC5AA, 0xB503, 0xE666, 0xE665, 0xBE23, 0x6D20, 0x7605, 0x7DC6, 0x6CA4, 0xEF95, 0xF756, 0xF755, 0xF714,
120    0xBD6D, 0xFFB6, 0xCDCC, 0xFF50, 0xB4C5, 0xDE2B, 0xF6D1, 0xC4EB, 0xFE52, 0xFE52, 0xD56D, 0xB468, 0xC4A8, 0xDCEA, 0xED2B, 0xFD2D,
121    0xDAE7, 0xB080, 0xD881, 0xE820, 0xF820, 0xF820, 0xF000, 0xD860, 0xB960, 0xF487, 0xF629, 0xAC82, 0xC589, 0x5220, 0x0020, 0x0000,
122    0x0000, 0x0000, 0x0800, 0x6AE3, 0xB525, 0xCDA3, 0xEEA5, 0xBDC2, 0x8D82, 0x7DC4, 0x75C4, 0x6CE4, 0xB60F, 0xF756, 0xFF56, 0xFF55,
123    0xBD6D, 0xDE6F, 0xE6CD, 0xFFAE, 0xDDE6, 0xDDE7, 0xD5CB, 0xBCE9, 0xE5AE, 0xC4A9, 0xC509, 0xFF91, 0xB426, 0xDD0A, 0xDD0A, 0xD429,
124    0xA1A2, 0xB942, 0xB8A0, 0xE963, 0xE820, 0xF020, 0xE060, 0xFAA5, 0xE3C5, 0xFE0A, 0xCD45, 0xACC5, 0xCDCD, 0x0800, 0x0000, 0x0000,
125    0x0000, 0x0000, 0x0000, 0x0800, 0xC548, 0xB4E1, 0xD604, 0xE6A6, 0xD6A7, 0x7CE0, 0x7583, 0x7565, 0x7C67, 0xFFB8, 0xFF56, 0xFF55,
126    0xBD4E, 0xD60D, 0xFFCC, 0xF727, 0xE603, 0xFF09, 0xEECB, 0x9C24, 0xDD6C, 0xC4E9, 0xFFEF, 0xD5E7, 0xB3E4, 0xE52C, 0xC428, 0xA2E4,
127    0xDB88, 0x90E0, 0xA1A0, 0xFD0F, 0xD020, 0xE020, 0xC0C0, 0xE344, 0xFE4B, 0xE628, 0x9C41, 0xD60B, 0x20A0, 0x0000, 0x0000, 0x0001,
128    0x0020, 0x0000, 0x0000, 0x0800, 0xA468, 0xBD25, 0xB4E2, 0xE686, 0xDE65, 0xAD82, 0x95C3, 0x7D24, 0x8468, 0xEF35, 0xFF76, 0xFF96,
129    0xC5AF, 0xA4A7, 0xFFA9, 0xF6E3, 0xE580, 0xD500, 0xEEC8, 0x8C00, 0xA424, 0xFF50, 0xEEC7, 0xBD42, 0xC446, 0xCC69, 0xABC6, 0xF5CE,
130    0xCB86, 0x8100, 0xFEB2, 0xB202, 0xC080, 0xC8C0, 0xEB85, 0xCC24, 0xFECA, 0x9C60, 0xC588, 0x83A4, 0x0800, 0x0000, 0x0000, 0x0000,
131    0x0000, 0x0020, 0x0000, 0x0000, 0x0800, 0xACE9, 0xBD45, 0xBD21, 0xEEC6, 0xD624, 0xB603, 0x6BE0, 0xF7B6, 0xEF36, 0xCE30, 0xEF12,
132    0xDEB2, 0xA4E7, 0xFFCB, 0xF643, 0xE4C0, 0xBBC0, 0xEEE6, 0xA4E0, 0xA482, 0xFF4C, 0xC5A1, 0xF6E8, 0xABC4, 0xB407, 0x9362, 0xFEAE,
133    0x71E0, 0xFE0D, 0xC344, 0x88E0, 0xA960, 0xDB44, 0xED47, 0xFF4B, 0xAC80, 0xACE3, 0xBD6C, 0x0820, 0x0820, 0x0000, 0x0020, 0x0000,
134    0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0840, 0xACE7, 0xBD24, 0xC522, 0xF708, 0xD645, 0xAD44, 0x7383, 0xB56D, 0xDECF, 0xB58A,
135    0xC60D, 0xD60A, 0xF686, 0xF5C2, 0xEC60, 0xCBA0, 0xEE04, 0xFFAA, 0xFFAC, 0xEEA7, 0xC580, 0xEEA8, 0xA3A4, 0x82A2, 0xF68C, 0xEE28,
136    0xFEEC, 0xE567, 0xAAE1, 0xC344, 0xC364, 0xDCA7, 0xFF2B, 0xA4A0, 0xA4C2, 0xDE6D, 0x1060, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
137    0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x20C0, 0x9427, 0xC568, 0xBCE3, 0xEE88, 0xE668, 0xB525, 0x7BA0, 0xAD24, 0xE6CB,
```

```
    0xFF6F, 0xFEEB, 0xFE45, 0xECE0, 0xDBA0, 0xE3E0, 0xE421, 0xDC41, 0xFDE4, 0xE5C2, 0xC4E0, 0xCD24, 0x6A40, 0xF690, 0xFECB, 0xF625,
    0xDDC2, 0xEE26, 0x8AE0, 0xBC24, 0xD527, 0xFEAC, 0xAC40, 0xACC3, 0xDE8F, 0x2100, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0001,
    0x0001, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0800, 0x1860, 0xCDCA, 0xBCE3, 0xE647, 0xEEE9, 0xB543, 0xC5C5, 0xEEC9,
    0xE648, 0xDD64, 0xDC80, 0xDB80, 0xFC43, 0xFBC4, 0xDA20, 0xCA20, 0xEC60, 0xED01, 0xB360, 0xCC63, 0xBC47, 0xFE2E, 0xE504, 0xBBE0,
    0xB460, 0xEE86, 0xEE8A, 0xBD26, 0xDDE9, 0x93E0, 0xB4C5, 0xC58A, 0x20C0, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
    0x0800, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0800, 0x0000, 0x0820, 0x2900, 0xC5A8, 0xACA1, 0xBD64, 0xEEEA, 0xBD04, 0xAC61,
    0xAC20, 0xCC61, 0xF462, 0xFC65, 0xF201, 0xD860, 0xE060, 0xF180, 0xE2C0, 0xD340, 0xF422, 0xFCC6, 0xF465, 0xEC24, 0xDB81, 0xB2C0,
    0xBCA1, 0xB523, 0xBD46, 0xACC5, 0x9C02, 0xCDAA, 0xCE2F, 0x0820, 0x0000, 0x0820, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020,
    0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x7343, 0xB527, 0xBD47, 0xA462, 0xCD87, 0xCD25,
    0xCD05, 0xBBC1, 0xE3C4, 0xC180, 0xD0C0, 0xE081, 0xE840, 0xD860, 0xEA82, 0xFC06, 0xEB02, 0xCA00, 0xD260, 0xEB64, 0xCAC1, 0xB2E0,
    0xAC02, 0x9C63, 0x83C2, 0xB508, 0xBD6B, 0x41A0, 0x0820, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000,
    0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0000, 0x0000, 0x0800, 0x0800, 0x2920, 0xC58B, 0xBD47, 0xAC63,
    0xB445, 0x6100, 0x8960, 0xA961, 0xB921, 0xC101, 0xC8C1, 0xC8E1, 0xB8E0, 0xB120, 0xB920, 0xB100, 0xA920, 0x9920, 0x8120, 0x8A40,
    0x93A3, 0xACC8, 0xC5AD, 0x62E3, 0x0820, 0x0020, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
    0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0800, 0x7302, 0xBD2A,
    0xBCCB, 0x5940, 0x8182, 0x80E0, 0x8080, 0x8860, 0x9060, 0x8820, 0x9060, 0x8840, 0x8820, 0x8880, 0x8080, 0x8122, 0x7161, 0x8AE5,
    0x72E4, 0x2900, 0x0820, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
    0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0800,
    0x49C4, 0x38C0, 0x7A46, 0x8A25, 0x9224, 0x91E3, 0x99E3, 0x99C4, 0x99E4, 0x99E5, 0x9A05, 0x9A26, 0x9206, 0x7A05, 0x5183, 0x1000,
    0x0800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0040, 0x0000, 0x0000, 0x0000, 0x0020, 0x0020, 0x0000, 0x0000, 0x0020, 0x0000,
    0x0000, 0x0000, 0x0000, 0x0020, 0x0000, 0x0000, 0x0000, 0x0001, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
    0x0800, 0x1000, 0x1800, 0x3000, 0x58E0, 0x6100, 0x68E0, 0x60C0, 0x4800, 0x3800, 0x4000, 0x3800, 0x2800, 0x2000, 0x1000, 0x0800,
    0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0020,
    };
```

Anexo 9. Construcción del prototipo

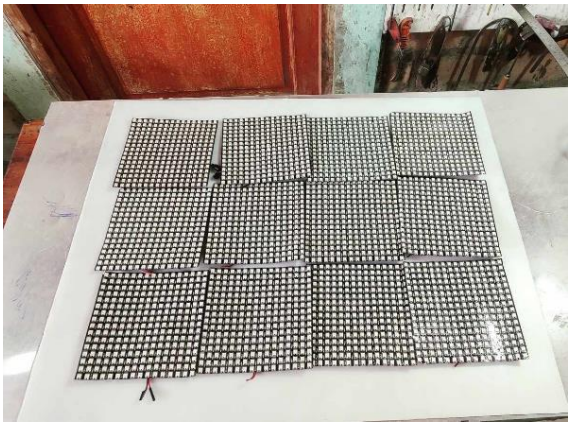


Figura 46. Proceso de construcción del prototipo.

Anexo 10. Certificación de la traducción del resumen.

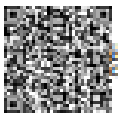
Loja, 22 de abril 2024

DAYANA ELIZABETH CARRION GUAMAN
LICENCIADA EN PEDAGOGIA DEL IDIOMA INGLES – UCE

C E R T I F I C O:

Que el documento aquí expuesto es fiel traducción del idioma español al idioma inglés del resumen del Trabajo de Integración Curricular titulado: "**Diseño y construcción de una pantalla informativa utilizando matrices LED RGB para proyección de texto**", de autoría de Joffre Fabian Iñiguez Quizhpe, C.I. 1105177248, de la Carrera de Ingeniería Electromecánica de la Universidad Nacional de Loja.

Lo certifico y autorizo hacer uso del presente en lo que a sus intereses convenga.



DAYANA ELIZABETH CARRION GUAMAN
LICENCIADA EN PEDAGOGIA DEL IDIOMA INGLES – UCE